Framework for Applications Authenticity Check

by

Rida Zojaj Naeem

A thesis submitted to the faculty of Information Security Department, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfilment of the requirements for the degree of MS in Information Security

March 2019

CERTIFICATE

Certified that final copy of MS Thesis written by **NS Rida Zojaj Naeem**, (Registration No. **00000206423**), of **Information Security Department, Military College of Signals** has been vetted by undersigned, found complete in all respects as per NUST Statutes / Regulations / MS Policy, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members and foreign/local evaluators of the scholar have also been incorporated in the said thesis.

Signature:
_____

Name of Supervisor

Assoc Prof Dr. Haider Abbas

Date:
_____

Signature (HOD):
_____

Date:
_____

Signature (Dean/Principal)
_____

Date:
_____

ABSTRACT

Installation and usage of different applications and softwares on different Operating Systems (OS) is a very common practice nowadays. The reason being easy availability of applications easy availability in Information Technology (IT) market. Moreover, these softwares also have user-friendly environment along with the various options they provide according to the needs of different organizations. Furthermore, most of the applications are free and the cost is negligible in case of paid softwares. Various critical organizations use such applications which makes their systems less complicated. The systems there may be secured and updated but installation of a vulnerable application may lead to a security breach.

With the advancement in technology, new threats are arising in the cybersecurity environment. These applications, with lots of visible options, may have some hidden vulnerabilities in them. The installation of such application may lead to making the system vulnerable to various attacks. It may result in business loss and disclosure of critical information.

Therefore, there is a need to check the authenticity of applications before their installation on critical systems. Different standards and guidelines are already available for checking the applications authenticity. Moreover, various frameworks and tools are also available for this purpose but those are very specific to an OS or a single point of check. Our thesis focuses on the critical study of such frameworks, tools, and identification of various options which should be a part of the application's authenticity check system. Moreover, an integrated framework has been proposed to check the application's authenticity before its installation. The framework covers the aspects of confidentiality, integrity, availability and authentication. Therefore, it guarantees the system security from vulnerabilities which may come along with the applications. The framework has been validated using five PDF readers applications. The data sets for validation of our framework has been collected from "CVE Details". This website provides vulnerabilities of specific application along with scoring or severity of the vulnerability.

DECLARATION

I hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

This thesis is dedicated to

MY BELOVED PARENTS

for their unconditional love, endless support and encouragement.

ACKNOWLEDGEMENTS

I am thankful to Allah Almighty who has bestowed me with the strength and the passion to accomplish this thesis. Without His consent, I could not have indulged myself in this task. I am grateful to my supervisor, Dr. Haider Abbas for his worthy supervision, constant support and valuable suggestions throughout this research. Also, I would thank my committee members Asst Prof Mian Muhammad Waseem Iqbal, and Lec Narmeen Shafqat for their valuable suggestions. Finally, I am highly thankful to my family and friends for their love and support.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background

Information and network technology have important roles in our infrastructure. Apart from their obvious advantage, technology has brought various dangers with itself.

Software and computing are an old concept. This goes back to ancient times with old devices like Abacus and being continued in this current technology age. An application or software is programmed instructions. These instructions are stored in the memory of digital computers for specific process execution. It is a basic part of our information technology age.

Initially, the softwares, for first generation computers, were stored in binary code i.e. instructions were written in binary code for mainframe computers. This was followed by the development of new modern software languages. The advancement in home computers was being done side by side. This advancement had greatly increased the scope and properties of available softwares. Starting with assembly language, it was followed by functional programming and object-oriented programming [1].

1.2 Software Security Assurance

Software Security Assurance [2] is a process to help in designing and development of software which may protect resources and data contained and controlled by it. The number of threats targeting critical systems is increasing along with emergence in technology. Therefore, the software which must be installed on critical systems should be assured with respect to security for business continuity and national security [3].

Software Security Assurance (SSA)is a process to ensure that software has been designed to operate at a specific security level, with no potential harm to the system on which it is being used. It starts with identification and categorization of information which will be contained or used by the software. The categorization is based on the sensitivity of the information. In the lowest category, the software with minimum security violation will lie and the top category will have software with high security violation impact. Categorization is followed by the development of security requirements including controls over systems, network, data management etc.

1.2.1 Causes of Security Problems

The vulnerabilities in software are a result of security bugs in it. It could possibly because of two reasons:

- Failure in satisfying the specified requirements or non-conformance

  It is not complicated e.g. an error in coding. It can be detected by validation and verification techniques and can be prevented through security assurance techniques. Applying these steps may improve the quality of software.

- Error in software requirements

  This occurs due to a wrong or inappropriate definition of software requirements according to system situation. These errors are difficult to identify as the software will perform as required during normal situation but may not act accordingly in some other state resulting in undesirable behavior.

1.2.2 Software Security Assurance Activities

Software security assurance activities may include security assurance and security requirements establishment. These are of two types:

- Some activities have their focus on the assurance that information possessed by the software has been properly categorized along with the implementation of proper security
- Some activities focus on the protection and control of software and its data and tools

A software security assurance program must ensure:

- Performance of security evaluation of software
- Establishment of security requirements for software, development, operation and maintenance processes
- Including security evaluation requirements during audit or review of software
- The existence of a proper process for providing security to software and ensure that any changes proposed in the software do not create vulnerabilities in the system
- Adequate physical security of software

Building in Security

Improving the development process of software will eventually increase its security. Different approaches are being used in this regard. One approach could be paying special attention to critical parts of software during its development. Many commercial-off-the-shelf (COTS) software packages are available which support assurance activities for softwares. Identifying and understanding the weaknesses in softwares will also affect software security. Common Weaknesses Enumeration Program [4] is a community-based project describing such weaknesses, but it's still is in its preliminary form.

Security Architecture analysis of software is another approach in this regard which involves four basic techniques.

- Logic Analysis: Evaluation of algorithms, equations and control logic
- Data Analysis: Evaluation of intended use of data and its description of the software
- Interface Analysis: Design verification of software's components interfaces with other system's components
- Constraint Analysis: Evaluation of software's design in comparison with restrictions imposed by real-world limitations and requirements including time, size, throughput, limitations of input, output, algorithm and equation, and other design limitations.

Software code analysis during development verifies that source code is correct, implements the required design and is not violating any security requirements. Source code review at user side may involve analysis in assembly language. Software code review is of three types:

- During an informal review, the developer selects one or more reviewers and provides them with the required material for review.
- The formal review is conducted at the end of the development phase and is conducted by the client.
- Inspection and walkthrough analysis is a detailed and a step-by-step review of software code to find errors.

Moreover, security testing of software includes penetration testing which confirms the design and security of software.

1.3 Testing Methods

Testing methods depend upon the procedure adopted for the purpose. There are many testing methods available, but the two main categories are black box testing and white box testing [5].

1.3.1 Black Box Testing

In this type of testing, the internal structure, implementation or design of the software is hidden from tester and is applicable to higher testing levels i.e. system or acceptance testing. Mostly independent softwares testers use this kind of testing. Programming and implementation knowledge are not a requirement for the tester.

1.3.2 White Box Testing

The tester has knowledge of design, implementation and internal structure of the software to be tested. White box testing is for lower levels of testing i.e. in the initial stages of software development. Generally, the softwares developers have the responsibility of white box testing. Implementation and programming knowledge is a requirement in this case.

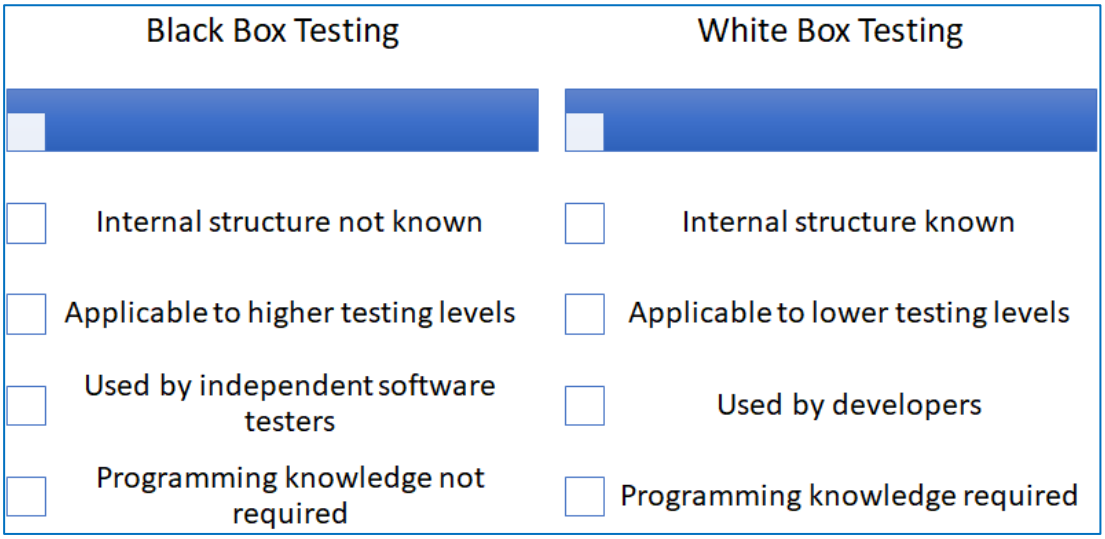

Figure 1 Comparison of Black and white Box Testing

A combination of these two types is known as Gray box testing. Some other forms of testing are manual and automatic testing.

1.3.3 Manual Testing

Since, various vulnerabilities are difficult to find through automatic testing method, therefore manual testing is used for this purpose. Specific skills and knowledge are required for this purpose. It may include business logic, code verification and design of software.

1.3.4 Automatic Testing

This technique uses other softwares and tools for scanning an application. A report is generated at the end of this process. No specific skills are required in this case.

Softwares and applications are being used in abundance in many government and private sectors. The reason being their easy availability and various options they provide. Moreover, they make the system less complex and easy to understand. These advantages make their use common, but this increasing technology has also brought some security issues with it.

1.4 Problem Statement

With the advancement in technological era, the threat vectors have also increased. Each one may have different way of attacking a system. One of the methods is attacking through a vulnerable software installed in the system. Softwares, without proper check, if installed in a critical system can be a threat to system security. These may have hidden vulnerabilities in them, which when installed make the system vulnerable to attacks. If the system contains critical information or a part of some critical infrastructure, it may result in loss of data or disruption in services. Moreover, the tools and frameworks available in this regard are either specific to a single aspect or are not freely available. Therefore, there is a need to check the authenticity of applications before their installation on critical systems.

1.5 Project Description

1.5.1 Objective

A framework for authenticity or credibility check for applications or softwares, to be installed, has been presented. This framework has four main pillars i.e. confidentiality, integrity, availability and authentication. There have been defined related parameters or vulnerabilities in all these four aspects. The framework will check an application on these parameters. After checking for these vulnerabilities in an application, the severity of probability of exploiting of vulnerability will be defined followed by average of all aspects which will give us a single value. Next is the decision of asset or application value. In the end, the business impact of the application will be decided. As a whole, we will get a single risk value which will lie in one of the four categories defined for credibility of application. The objective is to place an application in a category of credibility for organizations to decide whether to install the application or not.

Objectives of the thesis are as follows:

- Critical analysis of existing standards, guidelines and frameworks for applications authenticity check
- To propose and develop a framework for checking the authenticity of applications according to internationally recognized IT evaluation standards

1.5.2 Approach

Our approach has been shown in figure 2.

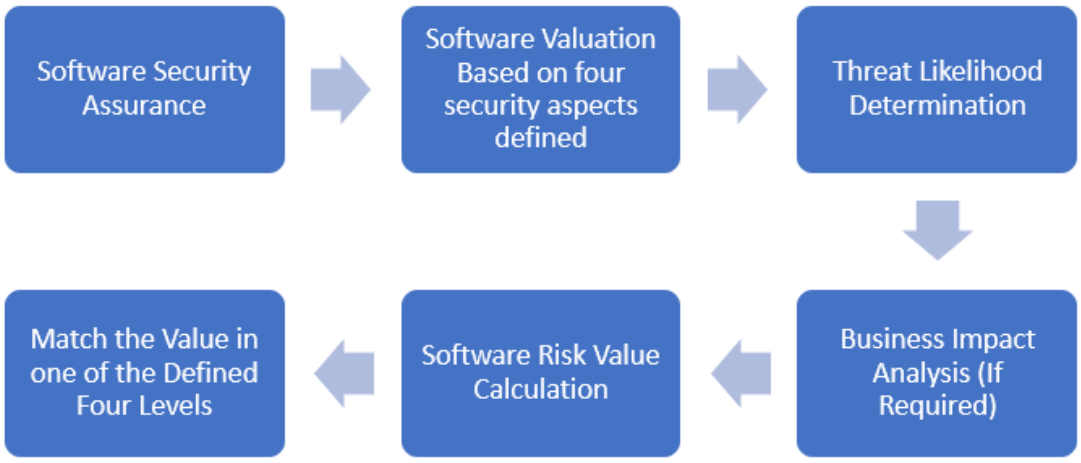

Figure 2: Approach of the Proposed Framework

1.5.3 Academic Objectives

- Understanding of Information Security Management Systems with respect to risk value calculations
- Analysis of various tools and softwares for applications authenticity check
- Understanding different frameworks and guidelines for applications security
- Analysis of different vulnerabilities of applications
- Analysis of various penetration tools and methods

1.5.4 Scope of the Project

The project covers the following areas:

- Identification of cyber-attacks applicable to softwares and applications with respect to confidentiality, integrity, availability, authenticity and non-repudiation
- Survey of tools verifying the integrity of softwares

- Survey of various tools for softwares credibility check
- Survey of various frameworks proposed for applications authenticity check
- Proposition of application authenticity check framework based on four security aspects
- Validation of proposed framework using a data set available on “CVE Details”

1.5.5 Areas of Applications / Advantages

This research will help:

- National Cyber Security Auditing and Evaluation Lab (NCSAEL) in securing their systems
- Military / Government / Private organizations in maintaining their security
- OS and Application developers for checking the security or risk level of their applications

1.6 Outline

This thesis has been divided into chapters:

Chapter 1: This chapter is an introduction to our thesis. It explains the background followed by an introduction of the proposed framework. Moreover, it also explains objectives and application areas of the thesis.

Chapter 2: It is a survey of various tools available for checking websites’ authenticity and software’s integrity.

Chapter 3: This chapter surveys different frameworks for assessments of softwares or systems.

Chapter 4: This chapter explains various paid systems, tools and methods provided by companies for security assessment of systems.

Chapter 5: This chapter proposes a framework based on four security aspects of confidentiality, integrity, availability, authentication and non-repudiation, for checking a software’s credibility.

Chapter 6: This chapter validates the proposed framework through a dataset available on “CVEDetails”. The software considered here is Adobe Acrobat 2017.

Chapter 7: This chapter is conclusion of the thesis with proposition of some future directions.

Chapter 2: Tools for Website/File's Credibility and Integrity Check

Different tools are available for checking softwares, websites and web applications credibility check. Each one offers different options. Integrity check is one of the main aspects in software authenticity checking, therefore, one can find a lot of option in this regard.

2.1 Easy Tools for Measuring Website Credibility [6]

When one is going to start a business online or any other use of websites, checking website's credibility becomes very important. There are some tools available which can be used to check website's credibility and size. It is recommended to use these tools in combination as none of these are authoritative enough to provide precise results.

2.1.1 Google PageRank

This tool is an algorithm developed by Google. It shows how important a website is according to Google. The number 0 to 10 ranks a website according to its importance like Facebook, Twitter has value 10 while other not so important websites may have value 0. We can get ranking of websites using Google Toolbar. Apart from these advantages, Google PageRank does not get updated frequently and may not provide information about the real number of visitors and content quality.

2.1.2 Alexa Ranking

It's a ranking system presented by Amazon and informs its users about visitors' traffic on a certain website. It ranks websites in ascending order according to the amount of traffic. Google has been assigned number 1 in this list. One can get this tool by signing up on www.alexa.com or installing its tool. The ranking is obtained through sampling and approximation; therefore, it can be distorted.

2.1.3 Web of Trust

Web of Trust (WOT) is a user rating-based ranking and shows trustworthiness of a website according to users. Its goal is protection of users from cyber scams. WOT tells about trust worthiness of a website and whether it's appropriate for children or not. This display is in the form of coloured metrics. It can be downloaded as extension in

your browser. Apart from this, sometimes there are no ratings available for smaller websites, so it is not being used as wieldy as other tools.

2.1.4 Moz Bar

It is a comprehensive tool telling about websites positioning according to page authority as well as its social media followers and other linked pages. The pages authority is analyzed in between the number 1 to 100 with higher score representing higher authority. One can get this tool from a browse's application store. Its free version only provides limited services.

2.1.5 Built With

It helps users to understand a technology which a website is using and its technical structure as well. It also tells about website's servers' location and other important information thus enlightening its users about website's credibility. The tool can be found at https://builtwith.com/. This tool may be too complex with not so familiar users in this domain.

2.1.6 Ghostery

This application tells about tools and plugins being used by the website. It is suitable for people who want to know about how much information of users is being collected by the website. One can make conclusions about professionality of the website. It can be found from its official website or a browser's application store. It is not an extensive tool as compared to other tools.

2.1.7 Qunatcast

This tool provides information about a website's audience e.g. demographics, interest, traffic size etc. The data is collected through a digital beacon and can be compared with other competitors. This tool can be found on their website. It can specifically be used for advertising purposes, therefore, the data collected is related to paid marketing activities.

2.1.8 Compete

It is an online tool which measures the traffic size of a website and compares with others. The paid version provides other information about mobile channel and channel effectiveness. Data collected is through user's behaviour. The limited free version only gives information about number of users visited the website.

2.1.9 SimilarWeb

It provides an overview of a website's traffic, search traffic, user's locations etc. The tool uses its own crawlers for this cause. The tool is available at http://www.similarweb.com/. Only the paid version provides enhanced features in this regard.

2.1.10 SEMrush

The tool collects information through Adwords, positions, rankings etc. It is suitable for marketing professionals. It can be obtained from its official website. Apart from this, it does not provide information related to link building.

All the tools discussed above have been summarized in Table 1.

Table 1: Summary of Website's Credibility Measuring Tools

Tool	Main Features
Google PageRank	shows how important a website is according to Google
Alexa Ranking	ranking system informs its users about visitors' traffic on a certain website
Web of Trust	user rating-based ranking and shows trustworthiness of a website according to users
Moz Bar	tells about websites positioning according to page authority as well as its social media followers and other linked pages
Built With	helps users to understand a technology which a website is using and its technical structure as well
Ghostery	tells about tools and plugins being used by the website
Qunatcast	provides information about a website's audience
Compete	measures the traffic size of a website and compare with others
SimilarWeb	provides an overview of a website's traffic, search traffic, user's locations etc
SEMrush	collects information through Adwords, positions, rankings etc

2.2 Comparison of File Verification Softwares [7]

Different softwares or applications are available, free or of cost, for checking the integrity of a file. The file, mostly, is analyzed based on the comparison of hash

calculated and hash available from an authentic source. Integrity is major part of checking the credibility of a software file therefore, its analysis holds importance. The tools, which could be used for this purpose are as follows:

2.2.1 AccuHash

This software is developed by Irnis Haliullin. The latest stable version of the software has been released in November 2008. The software is neither free nor open source. It is only available on Windows platform.

2.2.2 Beaver Simple File Verification (SFV)

It has been developed by Karl-Johan Sjögren. The latest version 1.2 is available in market free of cost but it is not available on open source platform. The software has been designed for Windows OS and its framework is written in.Net.

2.2.3 CDCheck

This software [8] detects, prevents and recovers the damaged files on CD-ROMs. It lays emphasis on error detection. One can ensure that data on a CD is safe and is not corrupted. This software has been developed by Mitja Perko. The latest stable version is 3.1.14.0 which has been released publicly in April 2008. It is not available free or on open source platforms. The software is available on Windows OS. CDCheck makes use of CRC32, MD2 and five more cryptographic functions for checking the file integrity.

2.2.4 Checksum Compare

Checksum Compare [9] has been developed by Sanktu Aire. The version 1.42 has been released February 2016 and is stable. It is available on Windows platform and framework is written in QT. It only uses MD5 and SHA-1 for checking file integrity. It is used for checking files and directories integrity.

2.2.5 Checksums

Checksums [10] is a shell extension library for adding a new page from properties windows of a file in File Explorer. It will show the checksums of the specific file selected. This software has been developed by Acera AS for Windows OS. Its latest free version 1.0.2 is available from December 2015 free of cost but it’s not available on open source platform. The framework is written in C++ language. It checks MD5, SHA-1 and three more for integrity check of a file.

2.2.6 CFV

This software [11] can be used to create and verify various formats of files including .sfv, .csv, .torrent files etc. This software has been developed by Matthew Mueller and its stable version 1.18.3 has been released in June 2009. It is a free software and available on open source platform. It has been designed for Linux, MAC and Windows OS and framework has been written in Python language. It calculates CRC32, MD5 and SHA-1 for checking integrity.

2.2.7 CHK

This software [12] calculates checksums of files, files duplicate etc. for checking the integrity. It has been developed by Ilya Muravyov. The latest stable version 2.35 has been released in November 2017. The software is free, but it is not available on open source platforms. It has been designed for Windows OS and framework is written in VCL. It supports different cryptographic algorithms i.e. CRC32, Tiger, Whirlpool Adler-32 and many other.

2.2.8 Cksum

Cksum [13] is a command for generating checksum in Unix-like OS. It assures that files from undependable source remain intact. It has been developed by POSIX. It has various stable versions. It is free of cost and available in open source. It is compatible with Linux and MAC OS platforms. It only calculates CRC32 for checking integrity of files.

2.2.9 CyoHash

CyoHash [14] is used within Windows Explorer as a Shell Extension. It has been developed by Graham Bull and its stable version 2.2.0.1 has been released in February 2016. Its free of cost and also available on open source platforms. It has been developed for Windows OS and its framework is written in C++ language. It checks the integrity of files by calculating their CRC32, MD5, and four others.

2.2.10 Data Digester

This software [15] is an integrated solution for checking checksums and OpenPGP signatures. It checks for errors in files and makes data exchange secure. Data Digester has been developed by Leisenfels and its stable version 1.6.15 was made available publicly by April 2014. It is a paid software and it is not available on open source platform. It is written in JAVA language and is compatible with Windows, Linux and MAC OS. It checks MD5, SHA-1, PGP and many others.

2.2.11 DeepDigest

This integrity checking software [16] calculates hashes of specified files and compares them with available one from trusted sources. Nick Stone developed the software and its latest stable version 1.3.27 as released in June 2014. It is free of cost and available on open source platform. Linux OS has the compatibility with this software. It has been written in QT. DeepDigest calculates MD5, SHA-1 and SHA-256.

2.2.12 DySFV

This software [17] is used for verifying the integrity of SFV files. It has been developed by Johan Sonesson. The latest stable version 1.4.0.0 of the software has been released in April 2013. It is available free of cost and on open source platforms. It is designed for Windows OS and written in .NET. The cryptographic algorithms used by the software are CRC-32, SHA-1, SHA-384, RIPEMD-160 and some others.

2.2.13 ExactFile

ExactFile [18] is a file verification tool. It checks the integrity of backup files, files copied to CD-ROMs i.e. files have not been changed and damaged. Brandon Staggs developed the software. Its latest stable version 1.0.0.15 has been released in February 2009. It is a free software but is not available on open source platforms. It runs on Windows OS and written in VCL. The cryptographic algorithms include are CRC32, Tiger, Adler-32, GOST, MD2 and some other.

2.2.14 eXpress CheckSum Calculator

It has been developed by Irnis Haliullin. The latest stable version 1.0 has been released in December 2002. It is a free software but not available on open source platforms. It has been designed for Windows platform.

2.2.15 FastSum

FastSum [19] is used for checking integrity of files transferred to external storage media. Kirill Zinov developed the software and the latest stable version is 1.7.0.403 GUI/1.9.0.149 CLI. Its command line edition is available free of cost, but standard edition is paid. It is not available in open source platforms and designed for Windows OS. It calculates MD5 of the files.

2.2.16 File Checksum Integrity Verifier (FCIV)

This software [20] has been developed by Microsoft which calculates hashes of files for integrity verification. It is a command prompt utility. The latest stable version has

been released in May 2004. It is free of cost but not available on open source platforms. It is only compatible with Windows OS. MD5 and SHA-1 are calculated by this utility.

2.2.17 Febooti Hash & CRC

Febooti Hash & CRC [21] is a simple tool for calculating various hash values for verifying integrity of files. It has been developed by Febooti software and its latest stable version has been available since September 2015. It is free of cost but is not available on Open source platforms. It is computable for Windows OS only. The cryptographic functions which it supports are CRC32, SHA-256, SHA-384, SHA-512, RIPEMD-320 and some other.

2.2.18 FileVerifier++

This software [22] calculates hashes and is a Windows utility for verifying integrity of files. Developed by Tom Bramer, the latest stable version 0.6.3.5830 has been released in May 2010. It is free of cost as well as available on open source platforms. It calculates hashes using CRC32, SHA-512, RIPEMD-128 and some other.

2.2.19 FlashSFV

FlashSFV [23] allows users to open and create SFV files and checks for any corrupt, bad or missing parts in the files. It has been developed by Charles DeWeese and its latest stable version 2.6 has been released in July 2007. It is free of cost and present on open source platforms. It is compatible with Windows OS. It supports CRC32 and MD5 algorithms.

2.2.20 FSUM

FSUm [24] is a fast file integrity checker. SlavaSoft developed the software with its stable version 2.02 released in March 2007. It is free of cost but not available on open source platforms. It has been designed for Windows OS. It has the capability to support different cryptographic algorithms.

2.2.21 Fsum Frontend

This software [25] is an easy to use tool for calculating checksums of files and text strings. It has the ability to handle multiple files at a time. It has been developed by Thierry Micholt. The latest stable version 1.5.5.1 has been available since May 2008. It is available free of cost as well as on open source platforms. Windows OS supports this software. It can support different algorithms with a total of 96 in number.

2.2.22 Power Shell

It has been developed by Microsoft with latest stable version 6.0 released in January 2018. It is free as well as available on open source platforms. It can be supported by Linux, MAC and Windows OS. The framework is in written in .NET Core and .NET framework. It calculates MD5, SHA-256, SHA-512 and three other cryptographic functions.

2.2.23 GTKHash

GTKHash [26] is a utility for calculating checksums. It can be redistributed and modified under the license of GNU General Public License. It has been developed by Tristan Heaven. The latest stable version 1.1.1 has been released in November 2017. It is free and open source software. Linux OS support this utility.

2.2.24 HashCalc

HashCalc [27] allows its users with calculation of checksums and HMAC of files, text and hex strings. It has been developed by SlavaSoft with latest stable version 2.02 released in March 2007. It is a free software, but it is not available in open source platforms. It is compatible with Windows OS. HashCalc can support total 12 cryptographic functions.

2.2.25 HashCheck Shell Extension

This software [28] has been developed by Kai Liu, various, gurnec and afterward by Software Development Laboratories (SDL). The version 2.4.0 was stable and was released in September 2016. It is a free software as well as available on open source platform. It is compatible with Windows OS. It can support CRC32, MD5, Sha-256 and three others.

2.2.26 HashForm

HashForm [29] is a Graphical User Interface (GUI) application to calculate and verify files or text hashes. Developed by Igor Gresovnik, the stable version of the software was 1.6. It is a free application, but it is not available on open source platform. It is compatible with Linux and Windows OS while its framework is written in .NET. It can calculate four cryptographic functions.

2.2.27 HashShell

HashShell [30] is same as HashForm except it is a command line software which contains commands for different cryptographic utilities. Apart from this, it is developed by Igor Gresovnik and its latest stable version is 1.6. It is a free application

but not available on open source platforms. Being compatible with Windows and Linux OS, its framework is written in .NET. Furthermore, it can calculate four cryptographic functions.

2.2.28 ilSFV

This software has been developed by Jud White. The latest stable version 1.10 of the software has been released in August 2011. It is a free application and well as available on open source platforms. It has been designed for Windows OS. Its framework is written in .NET3.5. It calculates CRC32, MD5 and SHA-1 cryptographic algorithms.

2.2.29 JDigest

This software [31] is free and open source calculator and verifier. It calculates MD5 and SHA-1 of graphical files. Its code is written in Java language and is compatible with Linux, MAC and Windows OS. Moreover, its version 0.3.0 is the latest stable version developed by gdassieu.

2.2.30 jHashCalc

A free and open source software developed by Death Master. The latest stable version 1.2 of the software has been released in October 2008. The framework is written in Java language. It has the ability to calculate MD5, SHA-512, RIPEMD-160, Whirlpool and five others.

2.2.31 Jacksum

Jacksum [32] is a free, open source and platform free software i.e. it is available on Linux, MAC and Windows OS. It has been developed by Johann N. Löfflmann. The version 1.7.0 is latest stable one and has been released in July 2007. This framework is written in Java Language. It can support various cryptographic algorithms with total of 58 in number.

2.2.32 MD5 File Hasher

This software [33] provides different functionalities i.e. calculations of MD5 hash, automatic scans of files and implementation of filters and exception on files for creation of Checkfiles. Digital Tronic developed the software. The latest stable version is version 1.5.0.1 and has been released in August 2018. It is a free software but is not available on open source platforms. Windows platform supports this software.

2.2.33 MD5 and SHA Checksum Utility

This software [34] allows its users to calculate and verify six types of hash functions of single or multiple files. It has been developed by Raymond Lin with latest stable

version 2.1. It is compatible with Windows OS. The cost is dependent on type of software i.e. it has various types of costs. Moreover, it is not available on open source platform.

2.2.34 Md5deep

This software [35] package is used in system administration, digital forensics and computer security. It can calculate checksums of large number of files using different cryptographic algorithms. It has been developed by Jesse Kornblum. Version 4.4 is the latest stable one and has been available since January 2014. It is a free of cost and open source software with compatibility with Windows, Linux and MAC OS. Moreover, the framework is written in C++ language. The algorithms which it supports are MD5, SHA-256, Tiger, SHA-1 and Whirlpool.

2.2.35 MD5sum

MD5sum [36] can calculate and verify 128-bit MD5 hashes. The Portable Operating System Interface (POSIX), which is a family of standards by IEEE Computer Society, developed the software. It has various stable versions. It is a free and open source software. Also, it can be supported by Linux, MAC and Windows OS.

2.2.36 MD5summer

This software [37] has been developed by Luke Pascoe for calculating MD5 hash of files for verifying their integrity. Its latest stable version is 1.2.0.11 has been released April 2006. It is available free of cost as well as open source software. Moreover, it has been designed for Windows OS. The framework is written in Delphi 7.

2.2.37 MD5 Win Verifier

MD5 Win Verifier [38] allows its users to check checksums of their downloaded files. This software has been developed by Serafino Sorleti. The version 1.1.0.5 is the latest stable version released in June 2014. It is a free software but is not available on open source platforms. It is only compatible with Windows OS while its framework is written in .NET. It can support MD5, SHA-1 and SHA-256 cryptographic algorithms.

2.2.38 PeaZip

It is a file archiver utility [39] on Open Source Technologies. Giorgio Tani developed the software with its latest stable version 6.6.1 released in October 2018. It is a free and open source software. Also, it is designed for Linux and Windows OS. Its framework is written in Free Pascal. Different algorithms are supported by this

software i.e. SHA-3, Whirpool, Adler-32, CRC16, CRC24, CRC64, eDonkey and MD4 and six others.

2.2.39 QuickSFV

This program [40] is a fast file verification program. It has been developed by Mercedes with latest stable version 2.36 released in March 2008. It is a free software but is not available on open source platforms. Moreover, Linux and Windows OS support this software. CRC32 and MD5 algorithms are calculated in this software.

2.2.40 RapidCRC

RapidCRC [41] is a Windows tool. It creates and checks checksums of files. Version 0.6.1 is the latest stable version and has been available since February 2005. It is a free as well as open source software. CRC32 and MD5 algorithms are supported by this tool.

2.2.41 RekSFV

This software [42] has been developed by TryAndError, Inc. Version 1.7.6 is the latest stable version available since July 2017. It is a free software but not an open source tool. It is only supported by Windows OS. The framework is written in VCL. It can support different algorithms including Tiger, MD4 and HAVAL and seven others.

2.2.42 RHash

This software has been developed by Aleksey (Akademgorodok). Version 1.3.3 is the latest stable version and has been available since August 2014. It is a free and open source software. Linux and Windows support this software and its framework is written in C language. It can calculate CRC32, MD5, SHA-512, RIPEMD-160 and many others.

2.2.43 Safe File Manager

Mindbytez created this software [43] for integrity verification of files. Its version 1.2 is the latest stable one has been released in August 2018. It is a free software but is not available on open source platforms. It is compatible with Linux and Windows and its framework is written in Java language. It can calculate MD5 hashes.

2.2.44 Sigcheck

Sigcheck [44] is a command line utility which can show timestamp information, version number and certificates of a file. Its developer is Mark Russinovich (Sysinternals - Microsoft). Its version 2.70 is the latest stable version released in May

2017. It is a freeware for Windows but is not available on open source platforms. It can calculate MD5, SHA-1 and SHA-256 algorithms.

2.2.45 SFV Checker

SFV checker [45] makes sure that files with multi-volume archives are complete and accurate using CRC32 algorithm. It notifies about any bad or corrupt part in a file. Traction Software developed the software with its latest stable version 2.01 released in August 2013. It is available on Windows with variable cost.

2.2.46 sha1sum

This software [46] calculates and verifies SHA-1 hashes of files for checking files integrity. POSIX developed the software has various stable versions compatible with Linux, MAC and Windows OS. It is a freeware as well as available on open source platforms.

2.2.47 sha3sum

It is also developed by POSIX for calculating SHA-3 hashes of files for integrity verification. It is a free and open source software as well as compatible with Windows, Linux and MAC OS. The language used for its development is Perl.

2.2.48 SuperSFV

Thomas Dixon developed this software which has a latest stable version 1.2. It is available free of cost but is not present on open source platforms. Moreover, it is only compatible with MAC OS.

2.2.49 TeraCopy

TeraCopy [47] is a freemium file transfer utility along with verification of integrity of files during their transfer. Code Sector is the software developer. Version 3.0 released in February 2017 is the latest stable version. The algorithms which it can support are CRC32, Panama, XxHash and many others.

2.2.50 Tom's Hash Explorer

This software has been developed by Tom Andreas Mannerud. Version 1.2.0.0 is the latest stable one and has been released in August 2016. It is a freeware but is not available on open source platforms. It has been developed for Windows OS and its framework is written in .NET. It can calculate six types of cryptographic functions.

2.2.51 Toolsley Hash and Validate

This software [48] is a web application which allows its users to calculate SHA-1, SHA-256, CRC32 and MD5 for the files in a browser. Toolsley.com developed the software and has various stable versions. It is a freeware but is not available on open source platforms. It is available on Windows, MAC and Linux OS.

2.2.52 Total Commander

Total Commander [49] is a file manager for Windows, Windows phone and Android OS. Christian Ghisler developed the software with its latest stable version 8.52a released in September 2015. It is a paid software and is not available on open source platforms. Its framework is written in Delphi. It calculates SHA-1, Sha-256, MD5, SHA-512 and CRC32.

2.2.53 WinHasher

WinHasher [50] is a free cryptographic hash calculator for Windows OS. Jeffrey T. Darlington developed the software with its latest stable version 1.6.2.67 released in August 2013. It is a freeware as well as open source software. Windows OS supports this software and the framework is written in .NET2.0. It can calculate eight types of cryptographic functions.

2.2.54 wxChecksums

It is a program [51] for calculating and verifying checksums. Julien Couot developed the software with latest stable version 1.2.2 released in March 2006. It is a freeware and opensource software. The software is available on Linux and Windows OS. Its framework is written GTK. It can calculate CRC32 and MD5 hashes.

All the tools discussed above have been summarized with respect to platforms they are compatible with and number of hash function they are able to calculate.

Table 2: Summary of Softwares' Integrity Checking Tools

Tool	**Platform**	**No. of CheckSums compatible**
AccuHash	Windows	-
Beaver Simple File Verification (SFV)	Windows	-
CDCheck	Windows	7

Checksum Compare	Windows	2
Checksums	Windows	5
CFV	Linux, MAC and Windows	3
CHK	Windows	22
Cksum	Linux and MAC	1
CyoHash	Windows	6
Data Digester	Windows, Linux and MAC	15
DeepDigest	Linux	3
DySFV	Windows	8
ExactFile	Windows	12
eXpress CheckSum Calculator	Windows	-
FastSum	Windows	1
File Checksum Integrity Verifier (FCIV)	Windows	2
Febooti Hash & CRC	Windows	15
FileVerifier++	Windows	12
FlashSFV	Windows	2
FSUM	Windows	12
Fsum Frontend	Windows	96
Power Shell	Linux, MAC and Windows	6
GTKHash	Linux	-
HashCalc	Windows	12
HashCheck Shell Extension	Windows	6
HashForm	Linux and Windows	4
HashShell	Linux and Windows	4
ilSFV	Windows	3

JDigest	Linux, MAC and Windows	-
jHashCalc	-	10
Jacksum	Linux, MAC and Windows	58
MD5 File Hasher	Windows	-
MD5 and SHA Checksum Utility	Windows	6
Md5deep	Linux, MAC and Windows	5
MD5sum	Linux, MAC and Windows	-
MD5summer	Windows	-
MD5 Win Verifier	Windows	3
PeaZip	Linux and Windows	14
QuickSFV	Linux and Windows	2
RapidCRC	Windows	2
RekSFV	Windows	10
RHash	Linux and Windows	10 and some others
Safe File Manager	Linux and Windows	1
Sigcheck	Windows	3
SFV Checker	Windows	-
sha1sum	Linux, MAC and Windows	1
sha3sum	Windows, Linux and MAC	1
SuperSFV	MAC	-
TeraCopy	-	13
Tom's Hash Explorer	Windows	6
Toolsley Hash and Validate	Windows, MAC and Linux	4

Total Commander	Windows, Windows phone and Android	5
WinHasher	Windows	8
wxChecksums	Linux and Windows	2

2.3 Conclusion

This chapter is a survey of various tools and softwares used for applications credibility check. The first part of the chapter described softwares used for websites credibility check. While the second part is about tools used for integrity check of softwares.

Chapter 3: Present Frameworks Available for Security Assessment

This chapter surveys different frameworks proposed for systems usability or security assessments. The first part of the chapter focuses on frameworks for checking usability of applications while second part focuses on security assessment.

3.1 Software Usability

3.1 1 Apps vs Devices

Subjective measures have been used in this paper [62] for usability evaluation of different mobile phone applications. The experiment has been conducted on two platforms i.e. OSX iPhone and O2 Orbit Windows phone. The aim was to understand the influence of devices on applications usability. Objective (examining the efficiency of applications) as well as subjective measures (examining user's feeling and thoughts) have been considered in experimentation.

The evaluation methods adopted included usability tests, focus groups and interviews. Various techniques proposed by Nielsen [63] were adopted. They included observations, questionnaire, interviews, focus groups and user's feedbacks. The applications under observation were CoPilot, a live satellite navigation application, and Mobile Google Apps, a set of applications by Google. The results of questioners and interviews showed that users were more satisfied with CoPilot application as compared with Google Apps. Another observation was that mobile phone features can greatly influence the usability of an application.

3.1.2 Usability evaluation of mobile applications

This study [64] is based on set of measures for evaluating the mobile phone's applications usability on different OS i.e. Android, Symbian and iOS. The objective is evaluation of a framework developed according to ISO 9126. 32 users took participation in this experiment. ISO 9041 and ISO 25062 have been used for objective measures. The applications under consideration were Google Apps and Google Maps. Authors highlighted some issues of mobile usability related to software and hardware. These issues must be taken in to account by developers to increase the quality of applications developed. The user satisfaction interaction questionnaire i.e. QUIS 7.0 has been used for accessing user satisfaction level. The results were analyzed and

indicated some challenges of hardware (screen size, resolution, storage) while using applications.

3.1.3 Usability Metric Framework

This work [65] surveys existing measurements models and metrics for desktop computing. From this study, authors built a set of selected guidelines. With the help of these guidelines, authors also presented a conceptual model for evaluating mobile phone applications. This model contains usability metrics for evaluation of mobile applications. This work has been divided into two phases. The first phase is study of previous metrics survey in this domain and finding the required guidelines. The second phase is development of usability evaluation metric. The main six guidelines were simplicity, time taken, safety, attractiveness, accuracy, features.

3.1.4 Usability Evaluation of SatNav Applications

This work [66] presents an examination of usability of mobile SatNav applications. The authors employed a mobile Goal Question Metric model (mGQM) for evaluating the usability of these applications. This is done by implementation of usability test for objective measures along with questionnaires and interviews of users for accessing the subjective measures. The results showed that usability problems of mobile SatNav applications are influenced by mobile phone unique features.

mGQM model contains usability metrics for accessing qualitative and quantitative measures of mobile applications. The goals of these metrics are simplicity, accuracy, time, features, safety and attractiveness. The authors used two SatNav applications for this purpose i.e. TomTom and CoPilot applications. The set of users were asked to install two applications and perform some tasks on them. Later, they were asked the mGQM questions followed by questionnaires and interviews. The results indicated that users were satisfied with the applications and found them useful. Moreover, some recommendations were enlargement of virtual keypad, optimization of tiny screen and increase the amount of voice assistance.

3.1.5 ISO/IEC 9126, Software Quality Standard

Mobile environment has many limitations i.e. energy, storage, heterogeneity, wireless network, mobility limited etc. This study [67] can help managers responsible for quality, to apply ISO 9126. It is a standard on software quality, specifically external quality model, for mobile environments. Limitations of mobile technology ha influence on software quality and it has been evaluated against the standard. The

influences have been first discussed and then aggregated for providing recommendations to managers, in this regard, so that they may evaluate their quality characteristics for mobile environment. This external quality is useful for evaluating reliability, efficiency characteristics and usability. This research work proposes a framework to deal with mobile environment limitations with respect to quality. This framework has three steps and has been developed for identification of quality characteristics according to ISO 9126.

ISO 9126 has four parts i.e. quality model, internal metrics, external metrics and quality in use metrics. The analysis process has two points; mobile limitations to be checked against external quality characteristics and influence of these limitations on software quality. The process has been divided into three main steps of analysis of external metrics, mobile limitations checklists vs. external metrics and calculations of degree of influence. The analysis process has been practically applied to six quality characteristics i.e. functionality, usability, portability, reliability, efficiency and maintainability. The degree of influence has been calculated through external metrics coverage rates of characteristics. The results were further analyzed, and recommendations were obtained through aggregated outcomes of the analysis.

The studies discussed above have been summarized in Table 3.

Table 3: Usability Frameworks Summary

Paper	**Platform / Applications Considered**	**Standard / Questionnaire / Method Used / Main Aspects**
Apps vs Devices	OSX iPhone and O2 Orbit Windows phone / CoPilot and Mobile Google Apps	usability tests focus groups, interviews and observations
Usability evaluation of mobile applications	Android, iOS and Symbian / Google Apps and Google Maps	ISO 9041 and ISO 25062
Usability Metric Framework	Mobile Phone Applications	simplicity, accuracy, time taken, features, safety and attractiveness

Usability Evaluation of SatNav Applications	Mobile Platform / SatNav Applications: TomTom and CoPilot	mobile Goal Question Metric model (mGQM)
ISO/IEC 9126, Software Quality Standard	Mobile Environment	ISO/IEC 9126

3.2 Software Security

3.2.1 Software Integrity Assurance

The authors [68] modified the Modula-2 development environment to provide cryptographic seals. These seals were required and exploited by CASS kernels and shells for providing integrity assurance of softwares. Authors first identified insecure features of Modula-2 language i.e. typecasting, pointer casting, variant records, variables export, export of non-opaque type and low-level features usage. They used secure gpm compiler for providing better assurance of integrity of softwares. Afterwards, the executables files were assured of integrity before their execution by their filenames, file's MD5 checksum and filename's MD5 checksum, file's MD5. Moreover, they also explained methods for developing secure softwares and suggested to distribute them using bespoke secure ftp protocol.

3.2 2 Kernel and Shell Based Applications Integrity Assurance

Verification of software integrity was an important issue in those years. Computer Architecture of Secure Systems (CASS) worked upon this issue by introducing integrity check at OS level. This paper [69] focuses on introduction of integrity check on kernel level on two different platforms i.e. Unix and SVR4.2 while Mach3.0 microkernel is the third platform considered.

The trusted compiler associated an unforgeable cryptographic hash and an IP address with executables and other files. The host OS must verify this signature before the execution of application. The verification is done by comparison of the digest calculated with the supplied one. The hash was computed using MD5 algorithm. Integrity of applications was verified through this process.

3.2.3 Monitoring of Distributed Component Interactions

This article [70] proposed a framework for monitoring of software components interaction. This framework is a part of a large framework built on the top of COBRA distributed processing environment. It has been developed to support the development

and testing of softwares. The framework takes Object Management Group (OMG) and Interface Definition Language (IDL) specifications as a contract for distributed interactions. It allows a tester to precisely monitor the software components interactions with expressive power, applicability and accuracy.

This framework specifically focuses on observing a systems' behavior but not representation of observed behavior in a specific format. The applications under consideration are target of this framework. A target may have different interactive components called entities. The proposed system observes softwares' behavior and produces monitoring events. The behavior of a software is interaction patterns between its different components. The proposed mechanism is flexible and configurable with emphasis on dynamic reconfiguration of monitoring output.

3.2.4 Software Security Assessment Instrument for Reducing Software Security Risk

This instrument [71] has been developed with the purpose of secure development of softwares i.e. security assurance of softwares during development phase. The instrument is a collection of various procedures and tools for supporting this objective. T is a formal approach which can be followed by developers. It includes a Vulnerability Matrix (VMatrix), based on Common Vulnerabilities and Exposures (CVE) listing, which has a specific application / platform and signature fields in its database. The base of databases of vulnerabilities is information of VMatrix. A set of Security Assessment Tool (SAT) is also included in this instrument. A property-based testing tool is included in SAT which looks for any vulnerability of software by slicing its software code. Moreover, this instrument also focuses on verification of softwares designs compliance according to some security properties using innovative model checking approach.

The approach followed is multifaceted with various prototype tools and activities categorized in different sub-domains i.e. assessment instrument for reducing risk, software's model-based development, model-based verification, security testing, validation and verification (V&V) techniques. Thus, this instrument consists of VMatrix, SATs, testing according to property-based method, security specification and verification according to model-based. These different parts can be used separately or in combination with various benefits i.e. less work to be done for

increased confidence in developed system, one tool with a lot of capabilities, identification of security properties and identification of some additional attacks.

3.2.5 Component-Based Systems

This paper [72] proposed a mechanism and its demonstration for maintenance and correlation of information of component-based applications about global causality. The authors developed a framework for understanding the behavior of such applications. The framework is a runtime monitoring mechanism for capturing semantic causality's execution behavior and propagation. Moreover, a characterization tool is also present for analysis and presentation of system behavior that has been captured.

The method followed actually exploits the source code of applications. IDL compiler has been used for components with IDL defined interfaces. A virtual tunnel, for facilitating the semantic causality information, is also constituted. The monitored behaviors may include CPU consumption, latency, resource utilization etc. In conclusion, this work included component level abstraction of applications, global causality capture and propagation. The unique features proposed by the system are integration of various techniques in this domain, end-to-end dynamic system topology capture, IDL compiler in its full version and capability of handling COM and COBRA applications.

3.2.6 Verification of Component-Based Software

This research work [73] focuses on the problem of evaluation of evolving software systems. It proposes a mechanism for identifying behavioral differences among various software versions. This may help in deducing possible unexpected interaction between different components of the software. The approach has been named Behavior Capture and Test (BCT), and has the purpose to combine run time monitoring, verification and testing. The run time monitoring collects data of single execution which is further used to derive behavioral characteristics of system and its components. These characteristics are then run time verified while single executions are used as regression test cases. BCT has also the capability to verify the integration of software components in a same or different system. BCT has five main phases i.e. generation and installation of behavior records, recording executions, distilling invariants, filtering behaviors and verifying invariants. The system proposed has many characteristics i.e. combining runtime verification, monitoring and test, reflection and

syntactic analysis, defining a mechanism for 'flattening' objects, application of computed invariants and preliminary experimental data to prove the validity of the proposed system.

3.2.7 Security Checking Based on SFT

This paper [74] focuses on optimization of fault feature database (FFD) and improving the checking efficiency of softwares faults. A new method based on Similar Feature Tree (SFT) has been proposed. The fault feature patterns of FFD were considered as nodes of SFT. An improved version of K-modes clustering algorithms and its associated rules were used in constructing SFT. During clustering, dissimilar objects of fixed number were selected as initial points thus construction of SFT is based on these clustering results. In improved K-modes clustering algorithm, the operation which previously chose initial cluster modes now ran as whole process of clustering. The process of checking through SFT is iterated according to "even left not right" rule, thus checking results are given by SFT. For experimental assurance, the categorical data from UCI Machine Learning Data Repository [75] was used while measure method [76] was used for experimentation. The results proved the proposed method to be superior than others in this regard.

3.2.8 Trustworthiness of Internet-based software

In this paper [77], authors have proposed a conceptual model for trustworthiness of internet-based software. Moreover, they also proposed an assurance framework virtual computing environment based on internet (iVEC). The proposed framework can deal with trustworthy properties of the software on capability, identity and behavior combined. The three core mechanisms of iVEC trustworthy assurance are authorization management for inter-domain computing environment, assurance on services availability and incentive mechanism design for collaboration. This framework can provide unified, trustworthy and transparent services to end-users and applications.

3.2.9 Formal security analysis in industry

The author [78] shared his firsthand experience in development of products, assessment of their quality and formal methods or tools for checking security of those products. Moreover, he also provided details of a project specifying and modelling Electronic Distribution Software (EDS).

This papers comments on the usage and impact of formal methods used in development industry including the evaluation and certification of products according to Common Criteria (CC). Practicing of formal methods specifically evaluation and certification methods have greatly influenced the quality of products. About EDS, its aim is to lessen the burden of softwares distribution specifically in modern airplanes. This has been done using physical disks till then. Moreover, it was under standardization ARINC 666 including Boeing and Airbus.

3.2.10 Model Checking Software Vulnerability Analysis

This paper [79] is a study of vulnerability analysis with respect to security based on model checking tools and formal methods. After thorough analysis, authors developed a Finite State Machine (FSM) model for formalization and reasoning about vulnerabilities. They have proposed an automatic model for verification that a particular program model (FSM) satisfies some specified security properties. This work bridges the gap between abstract specifications of security properties and their actual implementation. An illustrative analysis has also been performed for verifying the effectiveness of the proposed system through heap overflow vulnerability.

The system proposed is for analysis of software vulnerabilities through model checking, a famous method for verification of softwares and hardwares both. The model is based on FSM system for formalizing and reasoning about vulnerabilities. The process of exploitation has been decomposed in to small elementary activities. This makes the development of elementary FSM eFSM) model feasible due to simplicity of elementary activities. This eFSM then combined to develop a whole FSM. Several vulnerabilities analyzed in this regard were stack buffer overflow, heap overflow, integer overflow, file race conditions and format string vulnerabilities. The model checking has two parts i.e. first part is a state machine with defined variables, initial values or condition and second part is logic constraints over states. Authors used SPIN model checker. The steps followed were; encoding the software security in a finite state description, writing assertions in temporal logic and finally using model checker for verification of claims.

3.2.11 Source Code Model Checking Vulnerability Detection

In this research work [80], author used source code model checking technique to find out whether specified security guidelines were followed or not accordingly detect the related vulnerabilities. This checking applies model checking techniques to a source

code to check for any potential violation of security properties. Two SAP security guidelines related to Cross Site Scripting (XSS) and logging sensitive information were used as examples. Bandera Tool Set [81] as source code model checker in analysis of these two case studies. In the case of logging sensitive information, when a program has been developed, it is pre-processed and sent to Bandera with an auxiliary file for checking whether security guidelines for secure logging has been followed or not. Regarding XSS, auxiliary file and unexpected property specifications, the author was able to check whether the security guidelines were followed or not. In conclusion, author presented that how to use Bandera specification language for describing security programming guidelines with the help of two case studies.

3.2.12 Software Design Model Checking for Security Vulnerabilities

This research work [82] focuses on temporal security properties only. It describes a method of detection of security vulnerabilities in software design in UML through model checking tool. Definition of security properties is according to system requirements and they are formalized in formal language i.e. expressed in Linear Temporal Logic (LTL).

The authors presented an integrated technique for detection of design faults i.e. integration of model checking and system model language UML. If a software is in UML and security policy, the method proposed can extract the security properties and formally presents them in temporal logic language. With combination of security properties, the UML models are converted into PROMELA models i.e. input to SPIN model checker. SPIN keeps check on PROMELA model for satisfaction with LTL formulae. If it gets satisfied, software design is said to be free of faults otherwise, it may have design errors which may cause vulnerabilities. The results of the methods are either statically shown proving a security property of a model or gives alarm messages along with an execution path showing violation of properties.

3.1.13 Software Architecture Security Risk Analysis

Authors in this paper [83] introduced a novel approach for supporting security architecture analysis i.e. maintainability, usability, sustainability, and security and resilience against attacks, through various security scenarios and metrics. They formalized those scenarios and metrics signatures specifications through Object Constraints Language (OCL). Using these signatures, they checked target systems for any matches to these signatures for security scenarios and took measurements for

security metrics. New signatures and metrics can be incorporated with formal signature specifications. This approach targets definition of security scenarios and metrics at design, architecture and code levels. Furthermore, they also developed a system prototype tool for security analysis. This was the first approach in this domain which could support both metrics and scenario-based analysis. Authors validated their approach using attach scenarios defined in Common Attack Pattern Enumeration and Classification (CAPEC) database to evaluate signatures from NIST security principals.

This scheme can capture details of an attack scenario of a system including categories, consequences, preconditions and signatures etc. These signatures, when matched, indicates system vulnerability to the specified attack. The details are captured in a System Description Model (SDM) using UML. UML helps in capturing interrelations between different elements of system architecture. Security details are captured in a different Security Specification Model (SSM). Moreover, they mapped security entities to system entities. They also added signature evaluator and trade-off component for comparison between different architecture metrics with output as recommendations for software architecture. Thus, this scheme has five parts i.e. SDM, SSM, system security mappings, signature evaluation and trade-off analysis. Authors used Microsoft Visual Studio2010 UML modeler and Microsoft Visual Studio Modeling SDK for implementation of their scheme. Experimentation proved the scheme to be 90% precise.

3.2.14 Check Point Risk Evaluation Trust Model

A trust model of software behaviors has been presented [84]. It is based on check point risk evaluation with the aim accurate credibility evaluation of software behaviors. Authors first adopted fuzzy analytic hierarchy process (FAHP) for figuring out the weight of every risk factor of check point. Side by side, they assessed the modules values according to Markov Chain Usage Model. This step is for calculating the check points risk values and accumulating each risk check point. This is followed by adoption of rewarding and punishment mechanism for evaluation of software behaviors trustworthiness thus judging whether is software is credible or not. The experimentation showed that the proposed model effectiveness in distinguishing potential risks in softwares behaviors, evaluation of risks values and providing sufficient and reliable information for judging softwares credibility. The model

proposed has three features i.e. risk evaluation, monitoring softwares behaviors and accumulating every suspected risk check point followed by rewarding r punishment mechanism for evaluation of software.

3.2.15 Evaluation of Mobile Apps in Health Care

Mobile applications which are being used in e-health care system i.e. healthcare education or decision making, such application must be accurate and updated. Many such applications have not been regulated yet. For helping learners and users, this framework [85] has been proposed for evaluation of such applications.

For development of this framework, key components which are being used for website evaluation and evidence-based decision-making methods were surveyed and compiled in to a scheme. This scheme can be used for testing as well as evaluation of health and medical applications. Six criteria were selected for evaluation of applications i.e. accuracy, objectivity / bias, authority, timeliness / currency, usability, completeness / scope. This framework has been made available for self-learning and services.

3.2.16 Framework for Assessing Risk and Promotion of Safer Use of Medical Apps

This paper [86] highlights various types of risks which medical applications can contribute along with important contextual variables which may modify the specified risks. Moreover, it also signifies the need of risk assessment of medical applications for safer use. Authors also developed a framework comprising of usage scenarios, application complexity and contextual factors. This framework can be used to estimate the overall severity and probability of harm which may be caused by usage of a specific medical application.

Identification of different types of risks was the first step in developing the framework followed by understanding the key variables which may influence the medical applications. Authors also identified main external and inherent risk variables which were contributing to risks associated with medical application. The categories of framework were named as zone A, zone B and zone C with zone A to be safest applications to use while zone C to be least safe in this regard.

3.2.17 Security Threat Identification and Testing

This research work [87] presented a Security Threat Identification and Testing (STIATE) toolkit. The purpose is to support the development of softwares with respect to their security and vulnerabilities identification at early stages of their development.

It provides the services of threat modelling and analysis by automatic identification of threats during design and development time through model checking and mutation techniques. These techniques were incorporated on the top of sequence diagram with security annotations along with WHATIF conditions. STIATE can also identify threats during run time of applications by exploitation of identified threats and executing test cases. STIATE specifically provides different services i.e. usability, threat identification with translator, formal analysis and mutation techniques, and testing.

There are two main components of STIATE; a front end and a back end. The front end supports the security experts during security annotation process. The back end provides various functionalities i.e. obtaining XMI file as output from front end, its translation into formal specifications, followed by its analysis by SATMC model checker and in the end applying mutation engine for capturing security errors in web applications. Authors applied their system on an application scenario with Single Sign On protocol running in it. It showed that STIATE would have enabled DevTs detection of threats which may originate serious security flaws.

3.2.18 Multiple Layer Colluding Application Vulnerability Detection

Multilayer collision attack is a novel attack in information security domain with three parts; Spyware, Deputy and Delivery. Spyware steals data, pass it to Deputy which further passes it to Delivery thus helps in bypassing malware detection. This paper [88] proposes a mechanism for detecting both capability and deputy leaks. The process starts with decoding the .apk file to resource and assembly code. The mechanism proposed constructs correlation map from source data to intent through API calls thus finding any vulnerabilities in intent. For this purpose, there is need to trace potential function calls. Experimentation proved that deputy applications exist in Google Play.

Authors developed two models for tracing attached data of Android Intent. The first one is potential capability path tracer for identifying capability leak vulnerability. The second one is deputy path tracer for identifying deputy leak vulnerability. Deputy never uses API calls and have not got any permissions; thus, it is difficult to detect. Therefore, authors focused on privilege check of intent among communication between applications. This research work contributes in finding out deputy leaks, used sample attack to bypass private data and proved the existence of deputy applications. The process starts with APK configuration parser, correlation structure constructer, potential path matcher followed by APIs and URIs mapping permissions. Authors

developed an automatic download program which collected 8000 android applications which were analyzed for deputy and capability leaks. They found three applications with capability leak and ten with deputy leak.

3.2 19 Evaluation of Web Vulnerability Scanners

Authors [89] evaluated two open sources, multiplatform (Linux, Windows, OS X) and automated vulnerability scanners i.e. OWASP Zed Attack Proxy (OWASP ZAP) and Skipfish. The vulnerable applications which were put to experiment to these scanners were Damn Vulnerable Web Application (DVWA), a PHP / MySQL web application, and The Web Application Vulnerability Scanner Evaluation Project (WAVSEP).

Basically, vulnerability scanners have three parts i.e. a crawling component, an attacker component and an analysis component. Moreover, the two approaches to test applications are; white box testing i.e. source code analysis and black box texting i.e. application execution analysis. The different vulnerabilities for which these two scanners were analyzed on specified applications were SQL injection, Blind SQL injection, Reflected cross-site scripting, Persistent (stored) cross-site scripting, Local file injection, Remote file injection, Command execution and Cross site request forgery. When a specific vulnerability is shown executable by the scanner, authors counted it as a vulnerability. The results proved OWASP ZAP to be a better scanner than Skipfish but both of them are not perfect.

3.2.20 Security Quality Assurance

In this paper [90], authors surveyed various frameworks which can be secured at security level using infiltration testing. The survey included Pen testing approaches, methodologies, tools and security applied systems. Moreover, they also proposed an entrance testing technique for securing different frameworks like networks, databases, web applications and Android. There is a high reality powerlessness of entrance testing and no false positive exists about it. The proposed methodology starts with test plan followed by information gathering, breaking, exploitation, scanning and report. Moreover, authors also suggested that the pen testing can be divided in to two main categories of; breaking in to the system and securing the system.

3.2.21 Formal Specification and Verification of Security Guidelines

Authors proposed an approach [91] for integration of formal specifications and security guidelines verification at early stage of software development lifecycle. It is done by combination of model checking and information flow analysis. This work is

an extension of Labeled Transition Systems (LTS) by data dependence information for covering the end-to-end specifications and guidelines.

Authors are dealing with the problem that ho to check compliance of a software to some security requirements written in natural language. This requires formalization of informal written guidelines. The proposed work is actually providing means for expressing guidelines in a formal way. Moreover, the framework can also verify the adherence of a program to some specified guidelines and provides feedback of violation, if any. The framework is written in Java Programming Language. The idea proposed has the following parts; formal specification of security guidelines, development of Program Dependence Graph (PDG), information flow analysis (by JOANA tool) and verification. The framework provides analyzed programs' behavioral semantics in the form of a finite state machine called LTS.

3.2.22 Combining Control Flow Checking

The authors proposed a software-based method [92] for detection of control errors in embedded softwares, which may be caused by some transient faults or inculcated by attackers to gain privileged access. The idea is combination of Control Flow Checking (CFC) techniques for identification of errors. Authors focus is on CFC solutions i.e. Control Flow Checking for Safety and Security (CFCSS) having the advantage of no requirement of special hardware and high-performance rate. CFCSS checks after Control Flow Errors (CFE) has occurred. Moreover, checking instructions are also added at the end of basic block before the execution of another return instruction. Control Flow checking methods and CFCSS have been used in this paper.

3.2.23 Security Testing with Docker Containerization

This research work [93] describes the use of a provided automatic security testing service for developers to test the security of their java web applications. The system proposed contains Zed Attack Proxy tool for dynamic security scan, FindSecBugs plugins for static code analysis and OWASP dependency check tool for dependency check. The users must provide source code, database dump and URL (including credentials for login pages). System will run the scan providing security reports without false positives.

The aim of development of such system is to provide a fully extensible testing framework which may provide cloud-based services to developers for testing their web applications. The methodologies used in the framework are Docker Containerization,

configuration of the specified tools, automating deployment with Kubernetes, machine learning process for removing false positives, upload the source code and report generation for front end web applications followed by distributed computing solutions.

3.2.24 Vulnerability Assessment and Penetration Testing of Web Application

Open Web Applications Security Project Top 10 describes different attacks on web applications i.e. XSS, SQL injection, session exploitation, cross-site request forgery, buffer overflow, security misconfigurations, browser attack replay etc. Manual, as well as automatic penetration testing can be run on application depending upon the vulnerabilities. Comparative and collective analysis of these two tests / methods has been performed in this paper [94].

In automatic pen testing, lots of tools are available, which are open source as well as commercial products, having different functionalities. The comparison showed that manual testing can find almost all vulnerabilities of web applications which automatic may fail to do so.

3.2.25 Model-Checking Security of Java Software

In this paper [95], authors presented an approach which combines rule based static analysis and symbolic execution of Java code. The aim is to extract networks of timed automata from existing softwares followed by usage of Uppaal for their model checking against timed specifications. A case study has been practiced which applies the proposed methodology in a prototype tool. In this tool, the user specifies the program which he wants to abstract along with the set of predicates over the program variables. This real-world case study proved this approach to be helpful in model-checking security policies of Java software.

The research work discussed above have been summarized in the following table.

Table 4: Security Assessments Frameworks Summary

Paper	System / Applications Considered	Summary
Software Integrity Assurance	Computer Environment Softwares	Modification in Modula-2 development environment to provide cryptographic seals, integrity assurance of executable files with MD5 checksum
Kernel and Shell Based Applications Integrity Assurance	Unix, SVR4.2 and Mach3.0 microkerne	Introduction of integrity check on kernel level
Monitoring of Distributed Component Interactions	Computer Environment Softwares	a framework for monitoring of software components interaction and observing system behavior
Software Security Assessment Instrument for Reducing Software Security Risk	Computer Environment Softwares	The instrument is a collection of various procedures and tools for supporting secure development of softwares
Component-Based Systems	Component Based Applications	a mechanism and its demonstration for maintenance and correlation of global causality information
Verification of Component-Based Software	Computer Environment Softwares	mechanism for identifying behavioral differences among various software versions by combining run time

		monitoring, run time verification and testing
Security Checking Based on SFT	Computer Environment Softwares	optimization of fault feature database (FFD) and improving the checking efficiency of softwares faults, based on Similar Feature Tree (SFT)
Trustworthiness of Internet-based software	Computer Environment Softwares	a conceptual model for trustworthiness of internet-based software, also proposed an assurance framework virtual computing environment based on internet (iVEC)
Formal security analysis in industry	Development of products	Shared an experience in development of products, assessment of their quality and formal methods or tools for checking security of those products
Model Checking Software Vulnerability Analysis	Computer Environment Softwares	Development of a Finite State Machine (FSM) model for formalization and reasoning about vulnerabilities, also proposed an automatic model for verification that a program model (FSM) satisfies some specified security properties

Source Code Model Checking Vulnerability Detection	Computer Environment Softwares	used source code model checking technique to find out whether specified security guidelines were followed or not to find vulnerabilities
Software Design Model Checking for Security Vulnerabilities	Computer Environment Softwares	describes a method of detection of security vulnerabilities in software design in UML through model checking tool
Software Architecture Security Risk Analysis	Computer Software Environment	introduced a novel approach for supporting security architecture analysis through various security scenarios and metrics
Check Point Risk Evaluation Trust Model	Computer Software Environment	A trust model based on check point risk evaluation
Evaluation of Mobile Apps in Health Care	Smart Mobile Phone Environment	framework has been proposed for evaluation of mobile medical applications based on website evaluation and evidence-based decision-making methods
Framework for Assessing Risk and Promotion of Safer Use of Medical Apps	Smart Mobile Phone Environment	a framework comprising of usage scenarios, application complexity and contextual factors to estimate the overall severity and probability of

		harm which may be caused by usage of a specific medical application
Security Threat Identification and Testing	Computer Software Environment	presented a Security Threat Identification and Testing (STIATE) toolkit to support secure development of softwares from early stages
Multiple Layer Colluding Application Vulnerability Detection	Android Environment	Proposed a mechanism for detecting both capability and deputy leaks in applications
Evaluation of Web Vulnerability Scanners	Linux, Windows and OS X	Evaluation of two open sources vulnerability scanners i.e. OWASP Zed Attack Proxy (OWASP ZAP) and Skipfish
Security Quality Assurance	Networks, databases, web applications and Android	proposed an entrance testing technique for securing different frameworks
Formal Specification and Verification of Security Guidelines	Computer Software Environment	an approach for integration of formal specifications and security guidelines verification at early stage of software development lifecycle by combination of model checking and information flow analysis

Combining Control Flow Checking	Computer Software Environment	a software-based method for detection of control errors in embedded softwares by combination of Control Flow Checking (CFC) techniques
Security Testing with Docker Containerization	Web applications environment	describes the use of a provided automatic security testing service for developers to test the security of their java web applications
Vulnerability Assessment and Penetration Testing of Web Application	Web applications	Survey of penetration tools and methods
Model-Checking Security of Java Software	Java softwares	an approach which combines rule based static analysis and symbolic execution of Java code to extract networks of timed automata from existing softwares

Conclusion

This chapter explains different frameworks proposed for usability and security assessments of systems. The first part is of usability while second is of security.

Chapter 4: Proprietary Testing Services / Tools / Guidelines

This chapter surveys various testing services and tools provided by companies for security assessment of systems.

4.1 Security Testing Services of Veracode

Code security test involves analysis of code i.e. how it is written, its interaction with other parts of the system and identification of weaknesses which may lead an attacker to gain unauthorized access in the system. Veracode [52] provides extensive library for security testing of your organization. They provide helpful and informative resources for testing your web application security, network security, information security, software security etc. These solutions include tools i.e. application testing tools, code review tools, penetration testing tools, security review software, software testing tools, unit testing tools, vulnerability testing tools etc. Only demo versions of these tools are freely available that on providing some specific information to Veracode. Otherwise, one must buy these tools according to their requirement.

Application or software analysis is very important in securing an organization. Identification of vulnerabilities in application before its purchase and installation will help to prevent threats and negative impact posed by that software. It will ward off any loss of finance and reputation. Veracode has developed a cloud-based service for application testing. It will provide on-demand service of application security check i.e. finding security flaws in softwares developed internally or provided by third parties. Moreover, it also provides remediation reports thus help to prioritize flaws and fixes. This service can also help developers in testing the software and provides accurate and timely results. This platform combines static, dynamic and other manual testing techniques for checking the software security [53].

Another tool provided by Veracode is software testing tool. It specifically provides services for testing phase during software development lifecycle. It applies both static and dynamic analysis on software thus finding a malicious code or absence of a particular service in software which may leads to a security breach within the organization [54].

4.2 Oracle Database Security Assessment Tool

The Oracle Database Security Assessment Tool (DBSAT) [55] can be used to analyse a database configuration, users and their entitlements, security policies, identifies where sensitive data stays on your computer and helps to improve your organization posture with respect to security. It's a lightweight program which will not disturb the performance of the system.

The reports generated by DBSAT can used in:

- Fixing of immediate short-term risks
- Implementation of a comprehensive strategy with respect to security
- Providing support to some regulatory compliance program
- Promotion of best practices in security

4.2.1 Advantages of using DBSAT

- Quick and easy analysis of current security situation of your system, identification of sensitive data on your system along with its residing position
- Reduction in risk exposure through best practices provided by Oracle with respect to database security
- The security findings can further be used in compliance with your system with any international security standards
- Improvement in security posture of your organization database structure

4.2.3 DBSAT Components

The tool consists of three main components:

Collector

It performs SQL queries and run OS commands for collecting data from the system being analysed. The queries are applied on database dictionary views. The data collected is then written on JSON file that is used in next phase. The collector must be run on the server containing the database. Moreover, it must be run as an OS user with read permissions o directories and files.

Reporter

The data collected by collector is analysed by reporter and generates a security assessment report in different formats i.e. HTML, Excel, JSON and Text. The reporter can be run on any machine. It is not necessary to run reporter on the same system which is being anayzed.

Discoverer

It runs SQL queries and collected data from the system to be analyzed, on the basis of specified settings of configuration files. The queries are run on database dictionary views. A security assessment report is generated on the basis of data collected in HTML and CVS formats. It is not necessary to run the Discoverer on the same system which is being analysed. The data collected is then compared against specified patterns to find sensitive information. Discoverer must be a user with sufficient privileges.

Supporting System Requirements are as follows:

- Solaris x64 and Solaris SPARC64
- Linux x86-64
- Windows x64
- HP-UX IA (64-bit)
- IBM AIX (64-bit) & Linux on zSeries (64-bit)

The tool can be used on Oracle databases 10.2.0.5 and later released versions.

4.2.4 Output Files Security

The output files generated contains sensitive information about security weaknesses of a database. Therefore, proper surety guidelines should be implemented to prevent unauthorized access.

- The directories having these files must be secured with proper permissions
- Delete the files after applying the recommendations they have.
- Only encrypted form of these files must be shared
- Short term permissions should be granted

The DBSAT tool searches for sensitive data according to types defined in Pattern(s) file. It contains patterns for searching specific sensitive data type. Secure Socket Layer (SSL) or Transport Layer Security (TLS) can be applied between client and server.

The Discoverer must be configured accordingly for connection to server and collection of data. Database server shares a certificate with discoverer to establish the connection. The certificate is stored in wallet of Discoverer. The report may contain Title and Unique ID followed by status of database with “Pass” i.e. no error found and “Advisory” with recommendation to improve security posture. Moreover, it also contains summary, details, remarks and references.

4.3 Adaptive Security Appliance (ASA) Integrity Assurance

ASA Integrity assurance [56] describes the ways for verification of ASA software being run on device storage and memory has not been modified. Moreover, this document also describes the ways in protection against injection of malicious softwares in the system.

Currently, the attackers are focusing on attacks which may subvert the normal working of systems in an infrastructure. By compromising a system in an infrastructure e.g. firewall, attacker can gain privileged rights and may have access to sensitive data and other cryptographic materials. The malicious softwares used in the regard can be introduced in the Cisco system through the following ways:

- Alteration in original ASA software image that is stored on the file system of device
- Tempering of ASA memory during runtime
- Obtaining access of ASA Linux OS and installation of rootkit
- Modification in Bios firmware
- Combination of some or all of the above points

4.3.1 Potential Attack Methods

There are some methods to protect systems from code and memory manipulation. Administrators should verify the softwares and hardwares of the system must support the following features.

- Use of safe coding practices
- Use of digital signatures
- Use of secure boot

These features cannot protect the systems if the credentials have been compromised. Therefore, credentials protection must also be ensured by the administrator. Moreover,

attacker can modify the ASA software image by adding malicious code and load that code in a Cisco device which may support the modified code. This scenario can be implemented if the device loads OS from a writable device. Additionally, presence of a vulnerability in the system can also lead to system compromise by an attacker.

4.3.2 Identification Techniques

There are techniques available for identification of any modification done to ASA software image file and runtime memory.

Hash file validation of ASA software image file, this will calculate the MD5 or SHA-512 of image file loaded in memory. The calculated hash is then compared with hash provided by Cisco. It can only be used to check integrity of image file. Moreover, Cisco also digitally signs its image files to further strengthen the security. The authenticity of digitally signed files can be checked using **Show software authenticity file** command. The obtained value must match with certificate serial number value provided by Cisco.

The best way for verification of run time memory integrity is analysis of memory region called text. This section contains the executable code loaded in the memory. We can obtain the text section using core dump of the device. Administrator may have to force crash the ASA as currently there is no command for generation of core dump.

Additional indicators may include analysis of system log information on an external device. The Cisco ASA software can be configured to send its syslog information to an external system. Moreover, analysis of last time of system reload can provide additional information of any possible compromise. Important information in this regard could be found at system image file, uptime, configuration register and any modifications in configuration. Additionally, ROM monitor bootstrap program can provide valuable information about any compromise to the system. It can indicate any attempt to influence ASA boot sequence.

Failover events should also be analysed as an attacker may try to hide its activities under the cover of failover events. Failover history must be analysed regularly. The SSL VPN portal must be regularly checked for any compromise by attackers. The XML source code of SSL VPN must be analysed in this regard. Examples of SSL VPN plugins are remote desktop protocol | Route Discovery Protocol (RDP), Secure Shell (SSH), Virtual Network Computing (VNC), etc. The checksum of system

configurations must be kept in check and it should be ensured that only authorized personnel are making any changes in the configurations.

Best practices to maintain the integrity of Cisco ASA software are:

- Download the ASA software from official website of Cisco and then verify its SHA-512 hash value
- Copy the file to another storage media, set permissions to read-only and again verify its hash
- Remove the file from administrator system and relocate the file in the system which must use the software with any secure protocol implemented
- Again, verify the hash of software followed by configuration settings and loading the software in the memory

Change control mechanism must be implemented to ensure authorized changes in network devices. The server which distributes softwares in the network is of critical importance and must be hardened. Usage of best hardening applications, configurate of logging and auditing and placement of server in a secure network must be implemented in this regard. ASA software must be kept updated. Digitally signed ASA software images must be distributed, and secure boot of software must be ensured. Supply chain security must be completely ensured along with authentication, authorization and accounting. Authorization can be implemented through TACACS+ protocol for tight security. Credential management system must be enforced using TACACS+ and Radius protocol. Interactive management systems must be secured to prevent entrance of any malicious thing in the system. The system traffic can be visible through NetFlow version 9 which is supported by software. Traffic visibility will events visibility and data flow inside and outside the network. Furthermore, use of centralized and comprehensive logging must be ensured for further security.

4.4 Verify Software Integrity and Authenticity

Learn to use the GPG software to sign and verify a software [57] is a part of a whole course named as “Design and Analyze Secure Network System”. GPG software makes use of public/private key system for signing and verifying the software authenticity and source. The source can specifically verified by digital signatures using public / private key system. Moreover, SHA-1 and MD5 hashes are also calculated and compared with the available for further verifying the software.

4.5 Verification of authenticity of manually downloaded Apple Software Updates

If someone has downloaded a software or update package manually for MAC OS, it can be verified for authenticity and integrity [58]. Apple has digitally signed all the updates beforehand. If you download an update from MAC application store or Apple Support Downloads, you can verify them through their digital signatures. The steps for verification of completion and authenticity of an update are as follows:

- Open the installer and click the certificate or lock icon in the upper right corner of installer. If neither of the icons is present, the package has not been digitally signed so it should not be installed.
- Moreover, select the "Apple Software Update Certificate Authority". You can check for green mark of certificate here and certification authority.
- Check for SHA-1 value in details section. It should match with Apple current or earlier certificate.
- Continue with the installation by clicking OK, if package is faulty, installation will automatically stop.

4.6 OWASP Testing Guide

OWASP testing guide [59] has been presented for provision of guidance for performing penetration testing of web applications. It may include testing of identity management, error handling, cryptography and client size. It helps people to develop understanding that what, why, when, where and how to test web applications. Moreover, different organizations can use this framework for testing their web applications before their public release.

This testing guide covers different aspects of web application testing i.e. pre-requisites required for application testing and scope of testing, principles of successful testing, framework presented in relation with software development cycle and test for specific vulnerabilities. Furthermore, it also encourages organizations for relating cost with security required for an economical system.

The testing guide includes the following aspects:

Manual Inspections and Reviews:

- Manual inspections of processes, people, policies and technological decisions

- Require time and man power

Threat modelling:

- Risk assessment of applications
- Development of mitigation strategies for some common vulnerabilities

Code Review:

- Manual check of source code of application for any security issues
- Requires high skills in this specific domain

Penetration testing

- Remotely running application for finding security vulnerabilities
- Require proper permission from management

Moreover, a balanced approach must be followed in testing process. It must include several techniques covering manual reviews and technical testing. The Software Development Life Cycle (SDLC) must include definition, designing, scope, deployment and maintenance. The security test must include process review and manual inspections, code review and security testing.

The testing objective must be clear to the testers i.e. definition of metrics or number of vulnerabilities found according to which one can categorize an application. It must include validation of security controls and comparison of application with some standard security posture. Moreover, it may also include confidentiality, integrity and availability of data.

4.6.1 Developer's Security Testing

Developer must ensure that code being written is in accordance with best security standards being practiced internationally. Moreover, the libraries, methods etc. being used must be validated before their use in the code. Furthermore, secure code review must also be followed. A security test suite must include:

- Identity, authentication and access control
- Encoding and input validation
- Encryption
- Session and user management

- Error and exception handling
- Logging and auditing

4.6.2 Functional Tester's Security Testing

This security testing is performed during validation and integration phase. The objective is to validate the defence of application i.e. implementation of security controls at different layers by the application. It includes real attack scenarios using manual testing techniques and penetration testing tools. Moreover, the security test can be comprised of security vulnerabilities check list i.e. spoofing, information disclosures, buffer overflows, format strings, SQL injection and XSS injection, XML, SOAP, canonicalization issues, uploading of malicious files, denial of service and managed code and ActiveX controls (e.g., .NET).

4.6.3 OWASP Testing Framework

This testing framework has been developed for organizations to be used during software development life cycle. It is reference framework including techniques and tasks o be followed during SDLC. This framework is effective for the following phases of SDLC:

- Beginning of development
    - Properly define the life cycle of software development
    - Proper documentation including appropriate standards and policies
    - Development of measurements, metrics criteria and traceability must be ensured
- During definition and design
    - Review the security requirements which must include
        - User management
        - Authentication and Authorization
        - Data Confidentiality and integrity
        - Accountability and session management
        - Transport security
        - Triggered system segregation
        - Legislative and standards compliance
    - Design and architecture review
    - Design and review of Unified Modelling Language (UML) models describing application working

    - Develop and review threat models
- Development
    - Architecture and code walkthroughs
    - Multiple code reviews against a specific checklist including confidentiality, integrity, availability, technical exposures, any industry related specification, issues related to language and framework
- Deployment
    - Analysis of design and penetration testing
    - Testing of configuration management
- Maintenance and Operations
    - Conduction of operational management reviews
    - Monthly or quarterly health checks of the application
    - Ensure updates or any changes verification

4.6.4 Web Application Penetration Testing

Security testing is evaluating a computer or a network system and validating the effectiveness of security controls. Web application testing involves finding any weaknesses in the web application. Any issue found will be presented to the management for its proper mitigation. The testing methodology presented by OWASP is divided into two phases active and passive phases:

Passive Phase: The tester tries to find application logic and observes the application. He can use tools for information gathering e.g. observing the HTTP requests from the application.

Active Phases: The tester performs the methodologies defined in its sub-sections which are as follows:

- Information Gathering: Check for any configuration or information leakage by the application, fingerprinting of application architecture and identification of vulnerabilities
- Configuration Management: Observing and understanding the configurations of the server which is hosting the application and find any vulnerabilities there
- Identity Management: Testing of user registration process or weak user name policies

- Authentication: Involves testing of credentials transferred over an unsecure channel, default credentials, passwords policy etc.
- Authorization: It comes after authentication. The test includes for any bypass from authorization process, directory or files traversal and privilege escalation
- Session Management: Observing the user interaction with system, bypass from session management system, cookies, logout, timeout etc.
- Input Validation: Testing for SQL and XSS vulnerabilities and such other input validation errors
- Error Handling: In case of any error or exception, no information gets leaked even through error messages
- Cryptography: Test for weak SSL, TLS protocols, insufficient security, weak encryption and sensitive information being sent over an insecure channel
- Business Logic: Testing for any business logic flaws which may lead to bypassing of any security process
- Client Side Testing: Running the application at client side and check for any security flaws like XSS, HTML injection, resource manipulation etc.

4.7 MobileFirst application-authenticity validation

This tool [60] has been presented by IBM knowledge centre. MobileFirst is a whole big tool. The application authenticity check service is a by-product of this paid tool. This tool has been developed for protection against unlawful attempts to access information and fake applications. The process starts with deploying the MobileFirst tool on the server followed by creation of application-authenticity file. The security framework makes use of this file for validating the application authenticity. This security check is valid for Android, IOS and Windows OS.

4.8 Security Assessment of Corporate Information Systems in 2017

Kaspersky Lab Security Services department run different security assessment projects every year. A general summary and statistics of such assessment have been presented [61]. The aim is to provide information support to Information Technology (IT) security specialists regarding vulnerabilities and attack vectors in information systems. The projects consisted of external penetration testing, internal penetration testing and web applications security assessment. External penetration testing is assessment of security posture of an organization against external attacker while

internal penetration testing is assessment against internal intruders. Web application assessment is vulnerabilities and security flaws search in an application, which may be there due to some mistake during design and development phase.

Attack vectors which were able to penetrate the network perimeter were due to insufficient network filtering, public availability of network providing access to management interfaces, weak accounts passwords and web application vulnerabilities. Moreover, attack vectors were also successful because softwares in the systems were not updated.

Not much attention is paid on web applications security. According to the report Kaspersky lab [1], 73 % of attack vectors were successful due to vulnerabilities in the web applications installed in the systems. Different ways included were arbitrary file upload, which may lead to upload of command line interpreter and eventually provide access of the system to the attacker. Moreover, vulnerabilities like SQL injection, arbitrary file reading, XML external entity were used to gain sensitive information like passwords or hashes. Attacks were launched using accounts passwords through publicly available management interfaces.

Attackers gain access to critical systems through vulnerabilities in web applications and management interface that is available publicly following generally the following steps:

- Authentication bypass due to **SQL injection vulnerability** in web applications
- Passwords hashes access due to **sensitive information disclosure vulnerability**
- Easy offline password guessing due to **weak passwords**
- **XML External Entities Vulnerabilities** (access to only authorized or internal users) allowed the attacker to read files through credentials obtained
- Online password guessing through user names obtained in the attack due to **weak password and remote management interface available publicly**

- Functionality to record passwords, when entered by the administrators, is added along with “su” command, thus attacker gets the passwords when entered by the administrator

- Access obtained to internal network due to **Insecure Network Topology vulnerability**

4.8.1 Attacks through Management Interfaces

Such attacks mostly occur due to configuration flaws in the system. Passwords gained in such attacks were due to:

- Exploitation of vulnerabilities in the system like **Arbitrary File Reading vulnerability** may lead to access to clear text passwords
- Default credentials of CMS system, web applications and network devices
- Online passwords attack
- Access to another compromised system which may lead to credentials access. This is because use of same credentials on different systems

4.8.2 Internal Intruders Attack Vectors

Easiest attack vectors through which administrative privileges of a domain were obtained were:

- A combination of NBNS spoofing and NTML relay attacks which permit the attacker to access administrator’s NetNTML hash and eventually use it in authentication at domain controller.
- CVE-2011-0923 vulnerability exploitation in HP data protector and gain access to administrator password stored in the memory of Isass.exe process. This is because weak Single Sign On (SSO) implementation as some systems may store passwords in the memory.

The complex attacks in this domain were:

- No updated versions of network firmware devices with known vulnerabilities
- Weak passwords
- Same passwords on multiple systems
- NBNS protocol usage
- Accounts having extra privileges with SPN

The general steps in gaining administrator access of a domain are as follows:

- Vulnerability exploitation of D-link network storage web service. Execution of an arbitrary code is done with administrator privileges due the vulnerability. Creation of SSH tunnel for accessing management network as direct access has been prohibited by firewall rules. The vulnerability here is **obsolete software (D-link)**
- Detection of Cisco switch with default community string "Public" and an available SNMP service. IOS version of Cisco was identified by SNMP protocol. The vulnerability exploited here is **default SNMP community string.**
- Cisco IOS version information was used to find vulnerabilities specifically exploitation of CVE-2017-3881 (allows an attacker to execute an arbitrary code in Cisco IOS with maximum rights through Telnet protocol). Command interpreter access with maximum rights was obtained. Vulnerability is **obsolete software of Cisco.**
- Hash of local user's password was extracted.
- Followed by offline password guessing attack due to vulnerability i.e. **weak privileged user's rights**.
- Implementation of NBNS spoofing attack followed by interception of NetNTML2 hash due to vulnerability i.e. use of NBNS protocol.
- Again, offline password guessing attack on NetNTML2 hash due to weak user's password vulnerability
- A user account within the domain was used to implement Kerberoasting attack. Account TSG ticket with SPN was acquired.
- Local user account password obtained from Cisco switch was same as account with SPN. Vulnerability here is **password reuse and accounts with extra privileges**.

Most exploited vulnerabilities in 2017 were of obsolete softwares i.e. remote execution of code in Windows SMB i.e. MS17-010. Vulnerability analysis of web applications showed that most vulnerabilities exited at server side of applications and most common among them were exposure to sensitive data, functional level access control mossing, cross site scripting and SQL injection.

Conclusion

This chapter explains different systems, tools and methods proposed for security assessments of systems. These are all provided by various companies with some cost.

Chapter 5: Proposed Framework for Applications / Softwares' Authenticity Check

5.1 Introduction

Since softwares and applications are being used commonly nowadays, therefore their credibility check is an important aspect. Moreover, the frameworks proposed for this purpose are very specific to a point or are not free. Therefore, it is important to propose an all aspect authenticity check framework for this purpose. This framework will help in finding inadequacies present in a software with respect to security perspective. Moreover, it will also help in calculating the risk value of a software as an asset. This will help the system managers to decide easily whether to install a software on a critical system or not, based on its risk value. This work aims to propose confidentiality, integrity, availability and authentication aspects in the framework for softwares or applications authenticity check. The flow diagram of the framework is shown in Figure 3.

This framework parameters are based on Security Assessment of Corporate Information Systems in 2017 [61], OWASP Top 10 2017 [96] and other parameters found in the last some years. With respect to information security management, it is based on NIST 800-30 [97], ISO 27034 [98] and ISO 27002 [99]. Moreover, for software valuation purpose, the weightage of security aspects has been taken form Impact Metrics of security vulnerabilities [100]. Moreover, the CVSS [101] has been consulted for giving weightage to individual vulnerabilities.

5.2 Software Integrity

The first step of the framework is checking the integrity of the software or its patches. This can be done by calculating its hash and compare it with the hashes available from trusted sources. Different tools have been discussed in this regard in chapter 2. The recommended among them are CHK, Data Digester, ExactFile, Febooti Hash & CRC, File Verifier ++, Fsum Frontend, Hash Calc, Jacksum, PeaZip, RekSFV, RHash and Total Commander. These tools have been recommended based on the number of options these are offering as well as number of cryptographic hash function they can calculate.

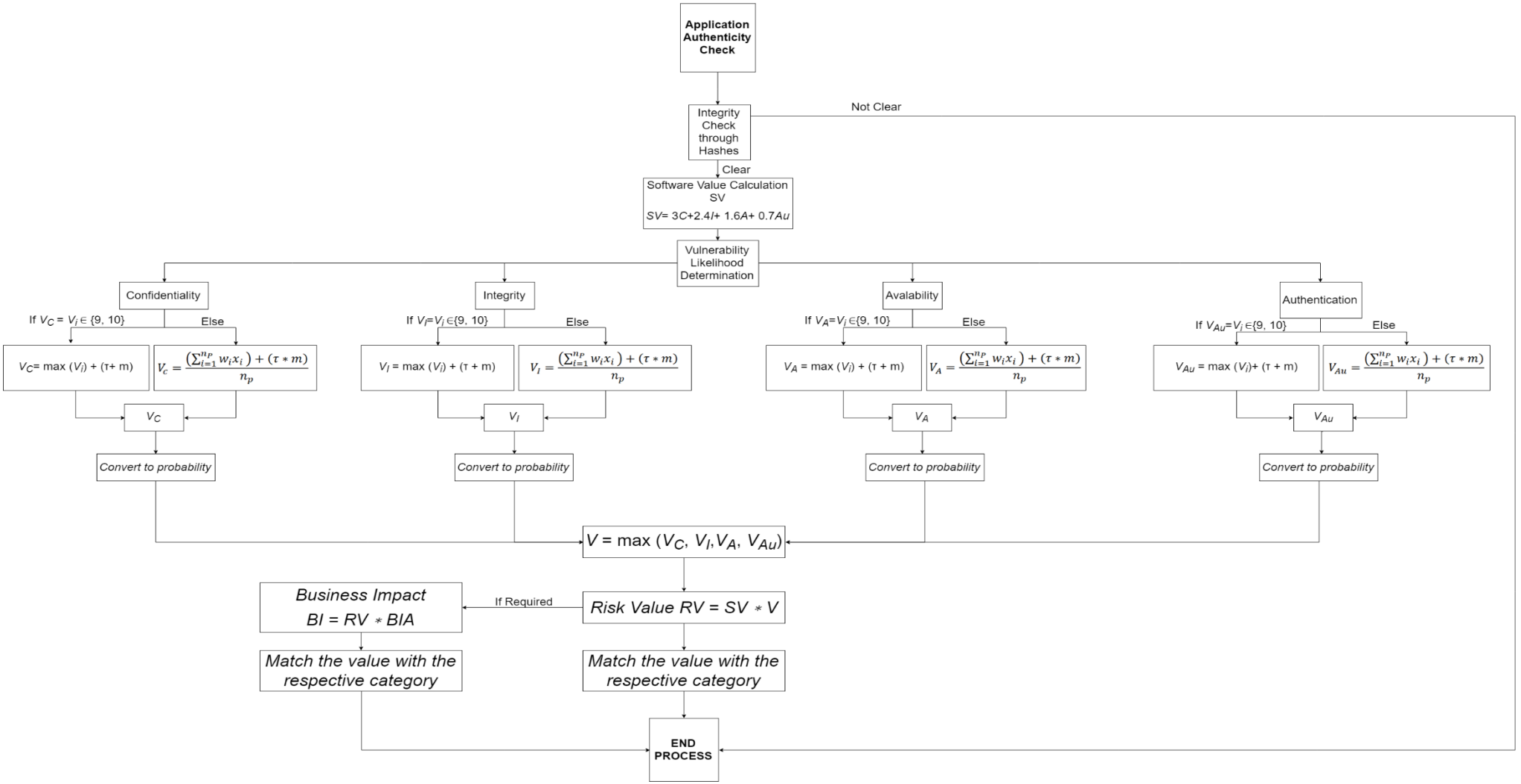

Figure 3: Flow Diagram of the Framework Proposed

Moreover, another check of whether the software is authentic or not. This can be done by setting a flag. If an organization receives a software and it must be checked whether it is from an authentic source or not. For that purpose, this framework sets a flag and check whether the software is authentic or pirated. It is done by calculating the hash and check whether this hash is available on some authentic source. If its available, then compare it with the hash calculated. If it matches, then one can move to next step of software authenticity check. Otherwise, the process must be stopped, and software must not be installed on critical system.

5.3 Software Valuation

Asset value is defined by looking at confidentiality, integrity, availability and authentication properties of the software. It is the value of what is the impact of any of the aspects breach of the software on the overall system.

$$SV = 3C + 2.4I + 1.6A + 0.7Au \qquad \text{(Eq. 1)}$$

Where

SV = Software Value

C = Confidentiality

I = Integrity

A = Availability

Au = Authentication

For example, if there is a confidentiality breach in the software, what will be its impact on overall system. If the impact is high, we will assign the highest value to confidentiality and conversely the lowest value. The highest value of an aspect is 3 while lowest value is 1.

High	3
Moderate	2
Low	1

With these values decided, the highest and lowest software are as follows:

SV (Max) = 3(3) + 2.4(3) +1.6(3) + 0.7(3) = 23.1

SV (Min) = 3(1) + 2.4(1) + 1.6(1) +0.7(1) = 7.7

5.4 Vulnerability / Parameters / Threat Likelihood Determination

This step is about deciding the likelihood of any aspect or threat present in the software. This is done by observing first the presence and then the probability of exploitation of a threat in the software. The vulnerabilities or parameters have been divided in four main categories of confidentiality, integrity, availability, authentication.

5.4.1 Confidentiality

Confidentiality is unauthorized exposure of data. It is the leak of private data which must be kept hidden.

Injection

Injection attack can be done when the user can input some untrusted data in the software or application. This may lead to execution of some unintended commands which may lead to confidentiality breach in the system. It can be done by queries, PHP queries, SQL, NoSQL, LDAP, OS commands, codes or URL manipulation.

Sensitive Data Exposure

Sensitive data exposure can happen if the application security is not proper. Attackers can sniff the data, and this may be because of use of no or weak encryption, use of weak TLS protocol or no use of SSL protocol. Loss of Logs data also included in this category.

Security Misconfiguration

Confidentiality breach can also occur if there are security misconfigurations in the software. These could be found in application database server and other options like weak or default passwords, default directories being used, error messages revealing the internal structure of the system, installation of all features of an application etc.

Memory Location

It should also be kept in check that what is the installation directory of the software and whether is it being installed in a directory or memory locations where it can cause confidentiality breach due to its other vulnerabilities. Moreover, it should be kept in check whether there is memory corruption being done by the software.

Ports Open / Close

If unnecessary ports are open, it can also cause confidentiality breach as data may be sent outside the system through these ports. These ports can also be open by a vulnerable application, so it should be check whether the application is opening unnecessary ports or not.

Database Protocol Vulnerabilities

Vulnerabilities in the database protocol of the software may also cause unauthorized access thus leading to confidentiality breach.

5.4.2 Integrity

Integrity is maintenance of consistency, accuracy and trustworthiness of data during its lifecycle. In our case, we will be considering the integrity of software file as well as integrity of system files.

Write to Original Code of Software

A software original code can be changed or formatted if it has an exploitable out-of-bound write vulnerability. It could lead to code execution according to the will of an attacker. Moreover, if an attacker can run an arbitrary code, it also comes under this category.

XML External Entries (XXE)

A software can be attacked upon XXE if it allows to upload malicious XML which may further exploit the vulnerable code or other dependencies. It can be used for stealing data, executing a code or performance of other malicious tasks. A software may be vulnerable to such attack if there is no check on upload of files.

Cross Site Scripting (XSS)

This attack can be performed if the software allows a user to insert untrusted data or scripts into a page of software. It may lead to stealing of data or defacement of application pages. XSS is of three types i.e. reflected, stored or DOM based. A software must be checked for all these types of XSS attacks.

Insecure Deserialization

Some of the softwares may save data on client side and they may be using object serialization. The softwares which are dependent on client for maintenance of data integrity may allow tempering of serialized data. This parameter is difficult to exploit.

Hashes of Critical Files

The hashes of critical files and folders are not maintained on a secure system. So, if there is some integrity breach it is difficult to find. Therefore, hashes must be maintained of critical files.

Application Access and Change

Applications access to directories must be checked as whether they are accessing critical files and making changes in them or not.

Man in the Middle Attack (MITM) / Session Hijack

In MITM or session hijack attack, the attacker secretly sniffs and may change the original message sent to or from the system. The software must use strong encryption to avoid such attacks.

5.4.3 Availability

It is the assurance that all systems, software as well as hardware are working properly and there is no disruption of services inside as well as outside an organization. The systems must be updated and upgraded for ensuring the availability of services.

DOS / DDOS Attack

The attacker seeks to make resources unavailable to users as well as clients. In DDOS attack, the attack is performed through zombies, i.e. attack is originated from other sources but not from the attacker machine. Buffer overflow, network flooding, Ping flooding, Smurf attack, SYN attack, illegitimate port scanning are forms of DOS or DDOS attack. The software must be checked for DOS or DDOS vulnerability.

Ransomware Attack

Ransomware attack is done through a malicious software which encrypts user's data and threatens to publish it unless the specified ransom is paid. The application or

software must be free of any vulnerability which may result in running the ransomware software.

Data Corruption

The software must not corrupt the systems data or other information. This check should be applied on the application or software.

5.4.4 Authentication

It is a process of identification of an individual through a username, password or some other digital identity. Privacy of data and users is maintained through proper authentication and access control. Broken access control may lead to irreparable loss.

Storage of Credentials / Hard-Coded Credentials

The software must be checked whether it stores credentials or not. Also, are those credentials were being saved in plain text or hashes. Specifically, hard-coded credentials must be checked in the code of the software. These aspects, if present in the software, make it vulnerable.

Session Cookies

The software must be checked for storage of session cookies and their encryption. These must not be stored in plain text.

Digital Signatures / Timestamps / Protected VPNs

Digital signatures or timestamps must be used for users or websites accessing the system from outside the organization. Otherwise, it makes the system vulnerable to attacks. This is also applicable to files downloaded by the software.

Each parameter under every aspect should be observed and check whether it is present or not. If some vulnerabilities are present out of all under one aspect, the only the vulnerabilities which are present must be considered. A threshold and error margin value are also calculated, and their product must be added in the vulnerability's likelihood determination equation. Moreover, the weightage given to any vulnerability is derived from the score categories of CVSS scoring. It is shown in Table 5.

$$\tau = 1 / N_L \quad \text{(Eq.2)}$$

$$m = N_L / (N_P + N_L) \quad \text{(Eq.3)}$$

where

τ = Threshold value

m = Error margin value

N_L = No. of levels = 4

N_P = No. of parameters under each aspect

N_P depends of each individual aspect

Table 5: CVSS Scoring and Weightage Values for SV

CVSS Category	Scoring Values	Weightage Value
Low	0.1 – 3.9	2
Medium	4.0 – 6.9	5.45
High	7.0 – 8.9	7.95
Critical	9 - 10	9.5

In vulnerability likelihood determination, we have three conditions:

- If any of the vulnerability does not exist under any aspect, then only the product of threshold and error margin value must be taken.

- If any vulnerability under any aspect lies under critical value according to score from National Vulnerability Database", then that aspect must be assigned the highest critical value of all the vulnerabilities present with addition of product of threshold and error margin value.

  $V_C = V_i \in \{9, 10\}$ then $V_C = \max(V_i) + (\tau + m)$ (Eq. 4)

  $V_I = V_i \in \{9, 10\}$ then $V_I = \max(V_i) + (\tau + m)$ (Eq. 5)

  $V_A = V_i \in \{9, 10\}$ then $V_A = \max(V_i) + (\tau + m)$ (Eq. 6)

  $V_{Au} = V_i \in \{9, 10\}$ then $V_{Au} = \max(V_i) + (\tau + m)$ (Eq. 7)

  where

  V_i = Vulnerability Likelihood of a security aspect i.e. C, I, A or Au.

- Otherwise, use the following formulas

$$V_c = \frac{\left(\sum_{i=1}^{n_P} w_i x_i\right) + (\tau * m)}{n_p} \quad \text{(Eq. 8)}$$

$$V_I = \frac{\left(\sum_{i=1}^{n_P} w_i x_i\right) + (\tau * m)}{n_p} \quad \text{(Eq. 9)}$$

$$V_A = \frac{\left(\sum_{i=1}^{n_P} w_i x_i\right) + (\tau * m)}{n_p} \quad \text{(Eq. 10)}$$

$$V_{Au} = \frac{\left(\sum_{i=1}^{n_P} w_i x_i\right) + (\tau * m)}{n_p} \quad \text{(Eq. 11)}$$

Where

w_i = weightage according to the scale

x_i = CVSS score of individual vulnerability

n_p = No. of vulnerabilities present

τ = Threshold value

m = Error margin value

Next is conversion of the calculated value to probability. This is done by dividing the value by 100. The probability value minimum would be 0.1 and maximum would be 1 i.e.

V (max) = 1

V (min) = 0.1

Moreover, if the $\boldsymbol{V_i}$ gets greater than 10 in the case where the any vulnerability scoring is 10, it must be rounded down to 100. After getting the probability value of each aspect, the final single value should be the maximum value of all the values of security aspects i.e.

$$V = \max (V_C, V_I, V_A, V_{Au}) \quad \text{(Eq. 12)}$$

5.5 Final Risk Value

Risk is the probability of an unwanted and / or unexpected event to occur.

The final risk calculation depends on Software Value SV and Vulnerability Likelihood V. The formula is as follows:

$$\text{Risk Value} = RV = SV * V \quad \text{(Eq. 13)}$$

$RV\ (max) = SV\ (max) * V\ (max) = 23.1 * 1 = 23.1$

$RV\ (min) = SV\ (min) * V\ (min) = 7.7 * 0.1 = 0.77$

Accordingly, the following scale has been defined for risk value of a software:

Table 6: Risk Values Scale Defined

Risk Value	**Risk Value Category**
0.77 – 5.9	Low
6 – 10.9	Medium
11 – 15.9	High
16 – 23	Critical

5.6 Business Impact Analysis

Business Impact Analysis (BIA) is performed to analyze the impact on business / reputation of an organization in case if the software gets down. It is observed based on time i.e. if the software gets down, how long an organization can go without it. A scale has been defined for this purpose as shown in the table:

Business Impact	**Value Assigned**
Low	1
Moderate	2
High	3

The business impact can be calculated with the following formula if required by an organization. It has not been added in the risk value of the software because of the reason that if the business impact analysis of a software falls in low category, its risk value is high and when it is used in the formula, it decreases the overall risk value of the software due to which a high risk software may falls in low category due to its business impact. It can be dangerous for the organizations with respect to security. The formula for the business impact calculation is as follows:

$$\text{Business Impact} = BI = BIA * RV \quad \text{(Eq. 14)}$$

Accordingly, the maximum and minimum value of BI is as follows:

BI (max) = BIA (max) ∗ RV (max) = 3 ∗ 23.1 = 69.3

BI (min) = BIA (min) ∗ RV (min) = 1 ∗ 0.77 = 0.77

The scale for this purpose has been defined as follows:

Table 7: Business Impact Scale Defined

Business Impact Value	Business Impact Category
0.77 – 16.9	Low
17 – 33.5	Medium
34 – 50.9	High
51 – 69.3	Critical

5.7 Defining the Risk Level

We have formalized four levels based on risk value of the software as shown in Table. These categories are based on exploitation of vulnerabilities with respect to critical infrastructure. They have been shown in the table 6.

Very Low (0.77 – 5.9)

It will be extremely difficult for a skilled hacker to exploit the vulnerability in the software. The software is not configured using common default practices. Moreover, the software design has been tested thoroughly. Physical access may be required to the attacker to exploit any vulnerability as it is difficult to find as well as exploit. Such softwares are well defined and implemented properly.

The softwares with very low risk values can be used by critical organizations, military and private organizations as well as personal use. Critical organizations can mitigate such risks by applying proper controls on their systems. For personal use, such softwares can be used as it is.

Medium (6 – 10.9)

Technical knowledge is required to exploit the vulnerabilities in the software. Programming and network knowledge may be required at some points. Some common practices may be used for software configuration. The software design has not been tested thoroughly. Physical access may be required to some extent for exploitation of vulnerabilities. Vulnerabilities are difficult to exploit if the attacker has no access to

any scanning and exploitation tools. Overall, such softwares are well defined to some extent.

Softwares having medium risk values should not be used by critical organizations. In case of private organizations, it can be used by applying proper controls while can be used as it is for personal use or with basic security configurations.

High (11 – 15.9)

Technical knowledge about scanning and exploitation tools is required for exploitation of vulnerabilities in softwares with risk level high. Programming and network knowledge may not be required here. Some default practices have been used for configuring the software. The software design has not been tested thoroughly. Physical access may not be required for exploitation of vulnerabilities. Vulnerabilities are easy to find and exploit if the attacker has proper knowledge of exploitation tools. Overall, the software strength and effectiveness has not been well defined.

Such softwares must not be used by critical organizations. For other private institution, controls must surely be put for their use. For personal use, it can be used with some advanced security configurations.

Critical (16 – 23)

Some basic technical and network knowledge is enough to exploit vulnerabilities of such softwares. Programming knowledge is not a pre-requisite in this case. Common or default practices have been used in software design. Physical access may not be required for exploitation of vulnerabilities. Vulnerabilities are easy to find and exploit using the exploitation tools. Overall, the software is not well defined and is a weak software.

Such softwares must not be used by critical as well as private organizations. It must also be not used for personal use to avoid any security breach.

Conclusion

The framework proposed have three main aspects in it; software value based on four security aspects, probability of exploitation of threats and business impact. After finding the risk value of the software using the formula in the equation, we can

categorize a software in one of the four risk value levels. These levels highlight some software properties along with recommendations on their use.

Chapter 6: Validation of the Proposed Framework

6.1 Introduction

The proposed framework has been validated using some data sets available on "CVE Details" about PDF Readers. These softwares include Sumatra PDF 2.2.1 2018 [102], Foxit Reader 9.4 2019 [103], PDF Xchange Editor 7.0.326.1 2018 [104], Nitro PDF 2.5.0.45 2018 [105], STDU Viewer 1.6.375 2017 [106] and Adobe Acrobat XI 2017 [107].

These have been developed for creation, manipulation, printing and managing files in Portable Document Format (PDF). The basic Reader is available for Desktop platform. They support annotating, viewing and printing of pdf files. These are freeware software and are commonly used. Their "Premium" services are available to users with paid subscription.

CVE Details [108] is an easy to use web interface with CVE vulnerability data. It provides information about different vendors, products, versions along with CVE entries, vulnerabilities, related to them. One can view statistics about vendors, versions and products. The data is taken from National Vulnerability Database (NVD) xml provided by National Institute of Standards and Technology. Some data is also collected from www.exploit-db.com.

6.2 Sumatra PDF 2.2.1 2018

It is a free PDF, ebook, XPS, DjVu, CHM, Comic Book reader for Windows. It is portable, small and powerful tool. According to CVE Details, it has one vulnerability:

CVE-2013-2830

Remote attackers can execute arbitrary code through a crafted PDF file by exploitation of this vulnerability. It falls under "write to original code of the software" vulnerability of Integrity security aspect.

Confidentiality	**Present / Not Preset**
Injection	×
Sensitive Data Exposure	×
Security Misconfiguration	×

Memory Location	×
Ports Open / Close	×
Data Base Protocol Vulnerabilities	×

Integrity	**Present / Not Present / Score**
Write to original code of the software	✓ 7.8
XML External Entries (XXE)	×
Cross Site Scripting (XXS)	×
Insecure Deserialization	×
Hashes of Critical Files	×
Application Access and Change	×
Man in the Middle Attack (MITM) / Session Hijack	×

Availability	**Present / Not Present**
DOS / DDOS Attack	×
Ransomware Attack	×
Data Corruption	×

Authentication	**Present / Not Present**
Storage of Credentials / Hard-Coded Credentials	×
Session Cookies	×
Digital Signatures / Timestamps / Protected VPNs	×

Software Value

Confidentiality

In the case of critical organization, if there is a confidentiality breach with respect to software i.e. disclosure of some critical pdf files, it is a critical matter for the organization therefore we assign the value of confidentiality 3.

Integrity

If there get some changes in the code of the software or content change of some critical files, then again it is a serious problem for that organization their integrity has been assigned the value 3.

Availability

If the software gets down due to some reason, then critical organizations must have a backup of the software or some other applications to read the files. Non-availability of the Acrobat software is not a big issue therefore, we assign it the value 1.

Authentication

Authentication is not involved for Sumatra PDF therefore, the value assigned is 1.

SV = 3(3) + 2.4(3) 1.6(1) + 0.7(1) = 18.5

Confidentiality	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (6 + 4) = 0.4$ $\tau * m = 0.25 * 0.4 = 0.1$ $n_p = 0$ $V_C = \tau * m = 0.1$
Integrity	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (7 + 4) = 0.36$ $\tau * m = 0.25 * 0.36 = 0.091$ $n_p = 1$ $V_I = (7.95 * 7.8 + 0.091) / 1 = 62.101$
Availability	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$ $n_p = 0$ $V_A = \tau * m = 0.1428$
Authentication	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$ $n_p = 0$ $V_{Au} = \tau * m = 0.1428$

$$V = \max (V_C, V_I, V_A, V_{Au}) \quad \text{(Eq. 12)}$$

V = max (0.1, 62.101, 0.1428, 0.1428) = 62.101

Converting to probability

V = 62.101 / 100 = 0.62101

RV = SV ∗ V = 18.5 ∗ 0.62101 = 11.49

Therefore, according to defined scale, this software lies in high risk category.

6.3 Foxit Reader 9.4 2019

It is a connected PDF and allows its users to comment on documents. It provides cloud-based services including PDF security, files management and PDF collaboration.

CVE-2019-5005

This vulnerability allowed DOS through image data. It lies in DOS attack parameter of Availability aspect. Score assigned is 5.5.

CVE-2019-5006

Null pointer dereference occurs during PDF parsing through exploitation of this vulnerability. It lies in Data Corruption parameter of Availability aspect. Score assigned is 5.5.

CVE-2019-5007

This vulnerability allows out of bound read information disclosure and may result in crash because of Null pointer dereference. It lies in Sensitive Data exposure parameter of Confidentiality aspect. Score assigned by CVSS is 7.1.

Confidentiality	**Present / Not Preset**
Injection	×
Sensitive Data Exposure	✓ 7.1
Security Misconfiguration	×
Memory Location	×
Ports Open / Close	×
Data Base Protocol Vulnerabilities	×

Integrity	**Present / Not Present**
Write to original code of the software	×
XML External Entries (XXE)	×
Cross Site Scripting (XXS)	×
Insecure Deserialization	×
Hashes of Critical Files	×
Application Access and Change	×
Man in the Middle Attack (MITM) / Session Hijack	×

Availability	**Present / Not Present**
DOS / DDOS Attack	✓ 5.5
Ransomware Attack	×
Data Corruption	✓ 5.5

Authentication	**Present / Not Present**
Storage of Credentials / Hard-Coded Credentials	×
Session Cookies	×
Digital Signatures / Timestamps / Protected VPNs	×

Software Value

Confidentiality

In the case of critical organization, if there is a confidentiality breach with respect to software i.e. disclosure of some critical pdf files, it is a critical matter for the organization therefore we assign the value of confidentiality 3.

Integrity

If there get some changes in the code of the software or content change of some critical files, then again it is a serious problem for that organization their integrity has been assigned the value 3.

Availability

If the software gets down due to some reason, then critical organizations must have a backup of the software or some other applications to read the files. Non-availability of the Acrobat software is not a big issue therefore, we assign it the value 1.

Authentication

If there is broken access control condition, it may lead unauthorized access to some options of the software which may lead to some loss. Authentication is required to login your account and use the features provided by the software therefore the value assigned 3.

SV = 3(3) + 2.4(3) 1.6(1) + 0.7(3) = 19.9

Confidentiality	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (6 + 4) = 0.4$ $\tau * m = 0.25 * 0.4 = 0.1$ $n_p = 1$ $V_C = (7.1 * 7.95 + 0.1) / 1 = 56.445$
Integrity	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (7 + 4) = 0.36$ $\tau * m = 0.25 * 0.36 = 0.091$ $n_p = 0$ $V_I = \tau * m = 0.091$
Availability	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$ $n_p = 2$ $V_A = (5.45 * 5.5 + 5.45 * 5.5 + 0.1428) / 2 = 30.0464$
Authentication	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$ $n_p = 0$ $V_{Au} = \tau * m = 0.1428$

$V = \max (V_C, V_I, V_A, V_{Au})$ (Eq. 12)

V = max (56.445, 0.091, 30.0464, 0.1428) = 56.445

Converting to probability

V = 56.445 / 100 = 0.56445

RV = SV ∗ V = 19.9 ∗ 0.56445 = 11.233

Therefore, according to defined scale, this software lies in high risk category.

6.4 PDF XChange Editor 7.0.326.1 2018

It is a small, fast and feature rich PDF viewer and editor. It allows its users to create, view, edit, annotate and digitally sign documents.

CVE-2018-16303

This vulnerability allowed DOS through crafted x:xmpmeta structure. It lies in DOS attack parameter of Availability aspect. Score assigned is 7.5.

Confidentiality	**Present / Not Preset**
Injection	×
Sensitive Data Exposure	×
Security Misconfiguration	×
Memory Location	×
Ports Open / Close	×
Data Base Protocol Vulnerabilities	×

Integrity	**Present / Not Present**
Write to original code of the software	×
XML External Entries (XXE)	×
Cross Site Scripting (XXS)	×
Insecure Deserialization	×
Hashes of Critical Files	×
Application Access and Change	×

Man in the Middle Attack (MITM) / Session Hijack	×

Availability	**Present / Not Present**
DOS / DDOS Attack	✓ 7.5
Ransomware Attack	×
Data Corruption	×

Authentication	**Present / Not Present**
Storage of Credentials / Hard-Coded Credentials	×
Session Cookies	×
Digital Signatures / Timestamps / Protected VPNs	×

Software Value

Confidentiality

In the case of critical organization, if there is a confidentiality breach with respect to software i.e. disclosure of some critical pdf files, it is a critical matter for the organization therefore we assign the value of confidentiality 3.

Integrity

If there get some changes in the code of the software or content change of some critical files, then again it is a serious problem for that organization their integrity has been assigned the value 3.

Availability

If the software gets down due to some reason, then critical organizations must have a backup of the software or some other applications to read the files. Non-availability of the Acrobat software is not a big issue therefore, we assign it the value 1.

Authentication

If there is broken access control condition, it may lead unauthorized access to some options of the software which may lead to some loss. Since, authentication is an optional aspect, therefore, we assign it a value of 2.

SV = 3(3) + 2.4(3) 1.6(1) + 0.7(2) = 19.2

Confidentiality	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (6 + 4) = 0.4$ $\tau * m = 0.25 * 0.4 = 0.1$ $n_p = 0$ $V_C = \tau * m = 0.1$
Integrity	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (7 + 4) = 0.36$ $\tau * m = 0.25 * 0.36 = 0.091$ $n_p = 0$ $V_I = \tau * m = 0.091$
Availability	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$ $n_p = 1$ $V_A = (7.95 * 7.5 + 0.1428) / 1 = 59.77$
Authentication	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$ $n_p = 0$ $V_{Au} = \tau * m = 0.1428$

$$V = \max (V_C, V_I, V_A, V_{Au}) \quad \text{(Eq. 12)}$$

V = max (0.1, 0.091, 59.77, 0.1428) = 59.77

Converting to probability

V = 59.77 / 100 = 0.5977

RV = SV * V = 19.2 * 0.5977 = 11.48

Therefore, according to defined scale, this software lies in high risk category.

6.5 Nitro PDF 2.5.0.45 2018

It enables its users to, quickly and easily, create, convert, edit, sign, review and protect PDF documents. Moreover, one can edit text and images, convert the PDF file to other formats and some other useful options.

CVE-2013-3553

An attacker can run arbitrary code through a crafted PDF file by exploiting this vulnerability. It lies under write to original code of the software parameter under Integrity aspect. The score assigned by CVSS is 7.8.

Confidentiality	**Present / Not Preset**
Injection	×
Sensitive Data Exposure	×
Security Misconfiguration	×
Memory Location	×
Ports Open / Close	×
Data Base Protocol Vulnerabilities	×

Integrity	**Present / Not Present**
Write to original code of the software	✓ 7.8
XML External Entries (XXE)	×
Cross Site Scripting (XXS)	×
Insecure Deserialization	×
Hashes of Critical Files	×
Application Access and Change	×
Man in the Middle Attack (MITM) / Session Hijack	×

Availability	**Present / Not Present**
DOS / DDOS Attack	×
Ransomware Attack	×
Data Corruption	×

Authentication	Present / Not Present
Storage of Credentials / Hard-Coded Credentials	×
Session Cookies	×
Digital Signatures / Timestamps / Protected VPNs	×

Software Value

Confidentiality

In the case of critical organization, if there is a confidentiality breach with respect to software i.e. disclosure of some critical pdf files, it is a critical matter for the organization therefore we assign the value of confidentiality 3.

Integrity

If there get some changes in the code of the software or content change of some critical files, then again it is a serious problem for that organization their integrity has been assigned the value 3.

Availability

If the software gets down due to some reason, then critical organizations must have a backup of the software or some other applications to read the files. Non-availability of the Acrobat software is not a big issue therefore, we assign it the value 1.

Authentication

If there is broken access control condition, it may lead unauthorized access to some options of the software which may lead to some loss. Since, authentication is an optional aspect, therefore, we assign it a value of 2.

SV = 3(3) + 2.4(3) 1.6(1) + 0.7(2) = 19.2

Confidentiality	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (6 + 4) = 0.4$ $\tau * m = 0.25 * 0.4 = 0.1$ $n_p = 0$ $V_C = \tau * m = 0.1$

Integrity	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (7 + 4) = 0.36$ $\tau * m = 0.25 * 0.36 = 0.091$ $n_p = 1$ $V_I = (7.95 * 7.8 + 0.091) / 1 = 62.101$
Availability	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$ $n_p = 0$ $V_A = \tau * m = 0.1428$
Authentication	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$ $n_p = 0$ $V_{Au} = \tau * m = 0.1428$

$$V = \max (V_C, V_I, V_A, V_{Au}) \quad \text{(Eq. 12)}$$

$V = \max (0.1, 62.101, 0.1428, 0.1428) = 62.101$

Converting to probability

$V = 62.101 / 100 = 0.62101$

$RV = SV * V = 19.2 * 0.62101 = 11.92$

Therefore, according to defined scale, this software lies in high risk category.

6.6 STDU Viewer 1.6.375 2017

It is a free software for viewing different file formats. The purpose is to view different formats using a single software. It supports TIFF, PDF, DjVu, XPS, JBIG2, WWF file formats.

CVE-2017-14692

He attacker can execute an arbitrary code by exploiting this vulnerability using a crafted .jb2 file. It lies under the category of Write original code of the software parameter under Integrity security aspect. The score assigned is 7.8.

CVE-2017-14691

The vulnerability can cause a DOS attack using a .crafted .jb2 file. It lies under DOS attack parameter under Availability aspect. The score assigned is 7.8.

Confidentiality	**Present / Not Preset**
Injection	×
Sensitive Data Exposure	×
Security Misconfiguration	×
Memory Location	×
Ports Open / Close	×
Data Base Protocol Vulnerabilities	×

Integrity	**Present / Not Present**
Write to original code of the software	✓ 7.8
XML External Entries (XXE)	×
Cross Site Scripting (XXS)	×
Insecure Deserialization	×
Hashes of Critical Files	×
Application Access and Change	×
Man in the Middle Attack (MITM) / Session Hijack	×

Availability	**Present / Not Present**
DOS / DDOS Attack	✓ 7.8
Ransomware Attack	×
Data Corruption	×

Authentication	**Present / Not Present**
Storage of Credentials / Hard-Coded Credentials	×
Session Cookies	×
Digital Signatures / Timestamps / Protected VPNs	×

Software Value

Confidentiality

In the case of critical organization, if there is a confidentiality breach with respect to software i.e. disclosure of some critical pdf files, it is a critical matter for the organization therefore we assign the value of confidentiality 3.

Integrity

If there get some changes in the code of the software or content change of some critical files, then again it is a serious problem for that organization their integrity has been assigned the value 3.

Availability

If the software gets down due to some reason, then critical organizations must have a backup of the software or some other applications to read the files. Non-availability of the Acrobat software is not a big issue therefore, we assign it the value 1.

Authentication

Authentication is not involved for STDU Viewer therefore, the value assigned is 1

SV = 3(3) + 2.4(3) 1.6(1) + 0.7(1) = 18.5

Confidentiality	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (6 + 4) = 0.4$ $\tau * m = 0.25 * 0.4 = 0.1$ $n_p = 0$ $V_C = \tau * m = 0.1$
Integrity	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (7 + 4) = 0.36$ $\tau * m = 0.25 * 0.36 = 0.091$ $n_p = 1$ $V_I = (7.95 * 7.8 + 0.091) / 1 = 62.101$
Availability	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$ $n_p = 1$ $V_A = (7.95 * 7.8 + 0.1428) / 1 = 62.1528$
Authentication	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$

	$n_p = 0$ $V_{Au} = \tau * m = 0.1428$

$$V = \max (V_C, V_I, V_A, V_{Au}) \quad \text{(Eq. 12)}$$

$$V = \max (0.1, 62.101, 62.1528, 0.1428) = 62.1528$$

Converting to probability

$$V = 62.1528 / 100 = 0.621528$$

$$RV = SV * V = 18.5 * 0.621528 = 11.50$$

Therefore, according to defined scale, this software lies in high risk category.

6.7 Adobe Acrobat XI 2017

Adobe 2017 is the latest perpetual version of Acrobat. It simplifies everyday PDF tasks and includes many productivity enhancements. It is used to create, edit and signs pdf documents. It can create PDF files from Microsoft Word, Excel and PowerPoint documents and vice versa. It can also change Photoshop (PSD), Illustrator (AI), or InDesign (INDD) files to pdf. We can also combine multiple files to single pdf. Moreover, it can also store and access files online securely with 20 GB storage.

CVE-2018-4918

It is an exploitable out of bound write vulnerability. It could write to original code of the software and could lead to arbitrary code execution with respect to a current user. It lies under the Write original code of the software parameter under Integrity security aspect. The score assigned s 9.8.

CVE-2018-4917

It is an exploitable heap overflow vulnerability. Successful exploitation may lead to arbitrary code execution. It will result in non-availability of pages of the software i.e. there will be a shutdown of affected resource and attacker can make the resource unavailable to legitimate users. It lies under DOS attack parameter of Availability security aspect.

CVE-2017-11307

It an exploitable out-of-bound read vulnerability. It refers to confidentiality breach. Exploitation of this vulnerability may lead to information disclosure resulting in

revealing of all system files. It lies under Sensitive Data Exposure of Confidentiality security aspect.

Confidentiality	**Present / Not Preset**
Injection	×
Sensitive Data Exposure	✓ 9.8
Security Misconfiguration	×
Memory Location	×
Ports Open / Close	×
Data Base Protocol Vulnerabilities	×

Integrity	**Present / Not Present**
Write to original code of the software	✓ 9.8
XML External Entries (XXE)	×
Cross Site Scripting (XXS)	×
Insecure Deserialization	×
Hashes of Critical Files	×
Application Access and Change	×
Man in the Middle Attack (MITM) / Session Hijack	×

Availability	**Present / Not Present**
DOS / DDOS Attack	✓ 9.8
Ransomware Attack	×
Data Corruption	×

Authentication	**Present / Not Present**
Storage of Credentials / Hard-Coded Credentials	×
Session Cookies	×
Digital Signatures / Timestamps / Protected VPNs	×

Software Value

Confidentiality

In the case of critical organization, if there is a confidentiality breach with respect to software i.e. disclosure of some critical pdf files, it is a critical matter for the organization therefore we assign the value of confidentiality 3.

Integrity

If there get some changes in the code of the software or content change of some critical files, then again it is a serious problem for that organization their integrity has been assigned the value 3.

Availability

If the software gets down due to some reason, then critical organizations must have a backup of the software or some other applications to read the files. Non-availability of the Acrobat software is not a big issue therefore, we assign it the value 1.

Authentication

In the case of premium subscription, the account login is required. If there is broken access control condition, it may lead unauthorized access to some options of the software which may lead to some loss. Since, authentication is an optional aspect, therefore, we assign it a value of 2.

SV = 3(3) + 2.4(3) 1.6(1) + 0.7(2) = 19.2

Confidentiality	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (6 + 4) = 0.4$ $\tau * m = 0.25 * 0.4 = 0.1$ $n_p = 1$ $V_C = (9.5 * 9.8 + 0.1) / 1 = 93.2$
Integrity	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (7 + 4) = 0.36$ $\tau * m = 0.25 * 0.36 = 0.091$ $n_p = 1$ $V_I = (9.5 * 9.8 + 0.091) / 1 = 93.191$
Availability	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$ $n_p = 1$

	$V_A = (9.5 * 9.8 + 0.1428) / 1 = 93.2428$
Authentication	$\tau = 1 / N_L = ¼ = 0.25$ $m = N_L / (N_P + N_L) = 4 / (3 + 4) = 0.57$ $\tau * m = 0.25 * 0.57 = 0.1428$ $n_p = 0$ $V_{Au} = \tau * m = 0.1428$

$$V = \max (V_C, V_I, V_A, V_{Au}) \quad \text{(Eq. 12)}$$

V = max (93.2, 93.191, 93.2428, 0.1428) = 93.2428

Converting to probability

V = 93.2428 / 100 = 0.932428

RV = SV * V = 19.2 * 0.932428 = 17.90

Therefore, according to defined scale, this software lies in critical risk category.

6.8 PDF Readers Comparison According to Risk Values

PDF Reader	**Risk Value**	**Category**
Sumatra PDF 2.2.1 2018	11.49	High
Foxit Reader 9.4 2019	11.233	High
PDF XChange Editor 7.0.326.1 2018	11.48	High
Nitro PDF 2.5.0.45 2018	11.92	High
STDU Viewer 1.6.375 2017	11.50	High
Adobe Acrobat XI 2017	17.90	Critical

The readers observed for validation of our framework, all lie under high or critical category according to the vulnerabilities present in them. Therefore, a critical organization must not install these softwares unless they have been patched or proper controls have been installed on the system.

Conclusion

This chapter shows the validation of the proposed framework using datasets available on CVE details about different PDF Readers. These readers, according to their vulnerabilities present in them, lie under high and critical risk values. The vulnerabilities present also have the scoring for high or critical category. Therefore, proper care must be taken before their installation on critical systems.

Chapter 7: Conclusion and Future Work

This thesis presents a survey of different tools for checking software security. Moreover, it also presents a study of different frameworks, systems or methods for checking softwares' usability and vulnerability. Moreover, it has proposed a framework for checking the credibility or authenticity of software or applications. The framework has six main steps:

- Software Integrity Assurance through checksums
- Software valuation based on four security aspects defined
- Determination of likelihood of exploitation of threat
- Business impact analysis
- Calculation of software risk value
- Match the software risk value to one of the four risk levels

The framework proposed will help in deciding institutions whether to install an application or not based on its risk value. If it is high, then it should not be installed on critical systems. If its low, then it can be used at all levels.

7.1 Future Work

The framework proposed is a theoretical study. In future, it can be implemented as a tool. The automatic tool, if developed, will be automatic and easy to use utility. Moreover, the following recommendations will help to mitigate the risks associated with vulnerabilities in softwares:

- All the data coming from softwares users must be checked.
- Access to sensitive data, directories and management interfaces must be restricted.
- Users must be provided with minimum permissions required and least privilege principle must be followed.
- Passwords minimum length and their complexity must be check along with implementation of frequent password change frequency. Moreover, elimination of using directory combinations of passwords must be ensured.
- Software and other components updates must be installed in a timely manner.

- Intrusion detection and prevention tools must be installed and functioning properly.
- Secure development of softwares must be ensured.
- Regular checks must be implemented on IT infrastructures and other cyber security applications.

Bibliography

[1] Paul E. Ceruzzi “A History of Modern Computing” 1998

[2] “Importance of Software Security Assurance”, https://www.oracle.com/corporate/security-practices/assurance/

[3] Karen Mercedes, Theodore Winograd "Enhancing The Development Life Cycle To Produce Secure Software" Archived 2012-03-30 at the Wayback Machine., Data & Analysis Center for Software, October 2008

[4] "Common Weaknesses Enumeration Project". Retrieved 26 August 2010.

[5] “Differences Between Black Box Testing and White Box Testing”, http://softwaretestingfundamentals.com/differences-between-black-box-testing-and-white-box-testing/

[6] “11 Easy Tools for Measuring Website’s Credibility”, https://www.userlike.com/en/blog/measure-website-credibility

[7] “Comparison of File Verification Software”, https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Comparison_of_file_verification_software.html

[8] “Welcome to CDCheck homepage”, http://kvipu.com/CDCheck/

[9] “Checksum Compare”, http://checksumcompare.sanktuaire.com/home

[10] “Checksums”, http://www.acera.no/products/checksums

[11] “cfv”, http://cfv.sourceforge.net/

[12] “CHK Checksum Utility”, http://compressme.net/

[13] “cksum”, http://pubs.opengroup.org/onlinepubs/9699919799/utilities/cksum.html

[14] “CyoHash”, https://sourceforge.net/projects/cyohash/

[15] “Data Digester Software”, http://www.leisenfels.com/products/digester

[16] “DeepDigest”, https://sourceforge.net/projects/deepdigest/

[17] “DySFV file verifier”, http://dysfv.sourceforge.net/

[18] “ExactFile”, https://www.exactfile.com/

[19] “FastSum Integrity Control”, http://www.fastsum.com/

[20] “Availability and description of the File Checksum Integrity Verifier utility”, https://support.microsoft.com/en-gb/help/841290/availability-and-description-of-the-file-checksum-integrity-verifier-u

[21] “Febooti Hash & CRC - freeware hash verification utility”, https://www.febooti.com/products/filetweak/members/hash-and-crc/

[22] “FileVerifier++”, https://sourceforge.net/projects/fileverifier/

[23] “FlashSFV”, http://www.trvx.com/flashsfv/

[24] “Fast File Integrity Checker, FSUM”, http://www.slavasoft.com/fsum/

[25] “FSUM Frontend”, http://fsumfe.sourceforge.net/

[26] “GTKHash”, https://gtkhash.sourceforge.io/

[27] “SlavaSoft HashCalc, HASH, CRC and HMAC Calculator, HashCalc 2.02”, http://www.slavasoft.com/hashcalc/

[28] “gurnec/HashCheck”, https://github.com/gurnec/HashCheck

[29] “HashForm”, http://www2.arnes.si/~ljc3m2/igor/software/IGLibShellApp/HashForm.html#hashform

[30] “HashShell”, http://www2.arnes.si/~ljc3m2/igor/software/IGLibShellApp/HashForm.html#hashshell

[31] “jdigest”, https://code.google.com/archive/p/jdigest/

[32] “Jacksum, Welcome”, https://project-jacksum.loefflmann.net/en/index.html

[33] “MD5 File Hasher, Create and verify checksums”, http://www.digital-tronic.com/md5-file-hasher/

[34] “MD5 and SHA Checksum Utility”, https://raylin.wordpress.com/downloads/md5-sha-1-checksum-utility/

[35] “md5deep”, https://github.com/jessek/hashdeep/

[36] “How to md5sum”, https://help.ubuntu.com/community/HowToMD5SUM

[37] “MD5summer”, http://www.md5summer.org/

[38] “MD5 Win Verifier”, https://www.microsoft.com/en-pk/p/md5-win-verifier/9wzdncrdcwlw?rtc=1&activetab=pivot:overviewtab

[39] “PeaZip”, http://www.peazip.org/

[40] “Official QuickSFV Homepage, Totally Useful Software”, http://www.quicksfv.org/index.html

[41] “Home of RapidCRC”, http://rapidcrc.sourceforge.net/

[42] “RekSFV”, https://www.kifoth.de/jane/misc/#RekSFV

[43] “mindbytez.com”, https://www.mindbytez.com/

[44] “Sigcheck v2.70”, https://docs.microsoft.com/en-gb/sysinternals/downloads/sigcheck

[45] “SFV Checker”, https://www.traction-software.co.uk/sfvchecker/

[46] “sha1sum”, https://lists.gnupg.org/pipermail/gnupg-announce/2004q4/000184.html

[47] “TeraCopy”, http://codesector.com/teracopy

[48] “Hash and Validate”, https://www.toolsley.com/hash.html

[49] “Total Commander”, https://www.ghisler.com/

[50] “winhasher”, https://github.com/gpfjeff/winhasher

[51] “wxChecksums”, http://wxchecksums.sourceforge.net/

[52] “Software Security Testing Tools”, https://www.veracode.com/security/security-testing-tools

[53] “Application Testing Tool for Web Application Analysis”, https://www.veracode.com/security/application-testing-tool

[54] “Software Testing Tools”, https://www.veracode.com/security/software-testing-tools

[55] “About the Oracle Database Security Assessment Tool”, Oracle Database, Security Assessment Tool User Guide, Release 2.0.2, E72220-05, July 2018

[56] “ASA Integrity Assurance”, https://tools.cisco.com/security/center/resources/asa_integrity_assurance.html

[57] “Verify Software Integrity and Authenticity”, https://www.coursera.org/lecture/design-secure-networked-systems/verify-software-integrity-and-authenticity-mRP6e

[58] “How to verify the authenticity of manually downloaded Apple Software Updates”, https://support.apple.com/hr-hr/HT202369

[59] “OWASP Testing Guide v4 Table of Contents”, https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents

[60] “Overview of MobileFirst application-authenticity validation”, https://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.dev.doc/dev/c_ibm_worklight_app_authentication_overview.html

[61] “Security Assessment of Corporate Information Systems in 2017”, Kaspersky for Business, August 2018.

[62] Azham Hussain, Maria Kutar, “Apps vs Devices: Can the Usability of Mobile Apps be Decoupled from the Device?,” IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 3, May 2012

[63] Nielsen, J.: „Usability Engineering“ (Morgan Kaufmann, 1994)

[64] Karima Moumane, Ali Idri, Alain Abran, “Usability evaluation of mobile applications using ISO 9241 and ISO 25062 standards,” Springer Plus, 2016

[65] Azham Hussain, Maria Kutar, “Usability Metric Framework for Mobile Phone Application,” Semantic Scholar, 2009

[66] Azham Hussain, and Maria Kutar, “Usability Evaluation of SatNav Application on Mobile Phone Using mGQM,” International Journal of Computer Information Systems and Industrial Management Applications, Volume 4, pp. 92-100, 2012

[67] Ali Idri, Karima Moumane, Alain Abran, “On the Use of Software Quality Standard ISO/IEC 9126 in Mobile Environments,” Proceedings Asia-Pacific Software Engineering Conference, APSEC, 2013

[68] George Mohay, S Y Chan, Tony Watson, “An Architecture for Software Integrity Assurance,” International Conference on Information, Communications and Signal Processing, September 1997

[69] George Mohay, Jeremy Zellers, “Kernel and Shell Based Applications Integrity Assurance,” Proceedings 13th Annual Computer Security Applications Conference, December 1997

[70] Nikolay K. Diakov, Harold J. Batteram, Hans Zendbelt, Marten J. van Sinderen, “Monitoring of Distributed Component Interactions,” In Workshop on reflective Middleware, New York, USA, 2000

[71] David P. Gilliam, John C. Kelly, John D. Powell, Matt Bishop, “Development of a Software Security Assessment Instrument to Reduce Software Security Risk,” Tenth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. WET ICE 2001, June 2001

[72] Jun Li, “Monitoring and Characterization of Component-Based Systems with Global Causality Capture”, 23rd International Conference on Distributed Computing Systems (ICDCS’03), 2003

[73] Leonardo Mariani, Mauro Pezze, “A Technique for Verifying Component-Based Software,” Electronic Notes in Theoretical Computer Science26, Elsevier, 2005

[74] Jiadong Ren, Lili Meng, Jiadong Ren, Changzhen Hu, Kunsheng Wang, “A New Method of Software Security Checking Based on Similar Feature Tree,” 1st International Conference on Information Science and Engineering, 2009

[75] UCI data repository http://www.sgi.com/tech/mlc/db/

[76] He, Zengyou; Deng, Shengchun; Xu, Xiaofei Approximation algorithms for K-modes clustering. Computational Intelligence International Conference on Intelligent Computing, ICIC 2006, Proceedings.

[77] WANG Huaimin, TANG Yangbin, YIN Gang, LI Lei, "Trustworthiness of Internet-based software," Science in China Series F: Information Sciences, Vol.49 No.6 759—773, 2006

[78] David van Oheimb, "Formal security analysis in industry, at the example of Electronic Distribution of aircraft Software (EDS)," Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, 2007

[79] Wang Chunlei, Huang Minhuan, He Ronghui, "Formally Analyzing Software Vulnerability Based on Model Checking," International Conference on Networks Security, Wireless Communications and Trusted Computing, 2009

[80] Keqin Li, "Towards Security Vulnerability Detection by Source Code Model Checking," Third International Conference on Software Testing, Verification, and Validation Workshops, April 2010

[81] J. Hatcliff and M. Dwyer, "Using the Bandera tool set to model-check properties of concurrent Java software," Proceedings of the 12th International Conference on Concurrency Theory, LNCS, Vol. 2154, Springer-Verlag, 2001, pp. 39–58.

[82] Li Jinhua, Li Jing, "Model checking security vulnerabilities in software design," 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), September 2010.

[83] Mohamed Almorsy, John Grundy, and Amani S. Ibrahim, "Automated Software Architecture Security Risk Analysis using Formalized Signatures," International Conference on Software Engineering, 2013

[84] Iunfeng Tian, Iianlei Feng, "Trust Model of Software Behaviors Based on Check Point Risk Evaluation," Third International Symposium on Information Science and Engineering, 2010

[85] Nicole Capdarest-Arest, "ABACUS: Developing a framework for evaluating mobile apps in health care," ARIZONA Health Sciences Library

[86] Thomas Lorchan Lewis, Jeremy C Wyatt, "mHealth and Mobile Medical Apps: A Framework to Assess Risk and Promote Safer Use," Journal of Medical Internet Research, vol. 16, issue 9, 2014

[87] Roberto Carbone, Luca Compagna, Annibale Panichella, Serena Elisa Ponta, “Security Threat Identification and Testing,” 8th International Conference on Software Testing, Verification and Validation (ICST), April 2015

[88] Ru Ko, Hahn-Ming Lee, Albert B. Jeng, Te-En Wei, “Vulnerability Detection of Multiple Layer Colluding Application through Intent Privilege Checking,” 5th International Conference on IT Convergence and Security (ICITCS), August 2015

[89] Yuma Makino, Vitaly Klyuev, “Evaluation of Web Vulnerability Scanners,” 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, September 2015

[90] Kamran Shaukat, Amber Faisal, Rabia Masood, Ayesha Usman, Usman Shaukat, “Security Quality Assurance through Penetration Testing,” 19th International Multi-Topic Conference (INMIC), December 2016

[91] Zeineb Zhioua, Yves Roudier, Rabea Boulifa Ameur, “Formal Specification and Verification of Security Guidelines,” 22nd Pacific Rim International Symposium on Dependable Computing, 2017

[92] Robert Gold, “Work-in-Progress: Combining Control Flow Checking for Safety and Security in Embedded Software,” International Conference on Embedded Software (EMSOFT), October 2017

[93] P.P.W. Pathirathna, V.A.I. Ayesha, W.A.T. Imihira, W.M.J.C. Wasala, Nuwan Kodagoda, E. A. T. D. Edirisinghe, “Security Testing as a Service with Docker Containerization,” 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), 2017

[94] Prof. Sangeeta Nagpure, Sonal Kurkure, “Vulnerability Assessment and Penetration Testing of Web Application,” International Conference on Computing, Communication, Control and Automation (ICCUBEA), August 2017

[95] Luca Spalazzi, Francesco Spegni, Giovanni Live, Martin Pinzge, “Towards Model-Checking Security of Real-Time Java Software,” International Conference on High Performance Computing & Simulation, 2018

[96] “Top 10-2017 Top 10” https://www.owasp.org/index.php/Top_10-2017_Top_10

[97] “Risk Management Guide for Information Technology Systems”, https://www.ucop.edu/information-technology-services/initiatives/resources-and-tools/sp800-30.pdf

[98] “ISO / IEC 27034”, https://www.iso.org/standard/44378.html

[99] “ISO / IEC 27002:2013” https://www.iso.org/standard/54533.html

[100] Georgios Spanos, Lefteris Angelis, “Impact Metrics of Security Vulnerabilities: Analysis and Weighing”, Information Security Journal: A Global Perspective, 2015

[101] “Common Vulnerability Scoring System v3.0: Specification Document”, https://www.first.org/cvss/specification-document

[102] “Sumatra PDF”, https://www.sumatrapdfreader.org/free-pdf-reader.html

[103] “Foxit”, https://www.foxitsoftware.com/pdf-reader/

[104] “PDF-XChange Editor”, https://www.tracker-software.com/product/pdf-xchange-editor

[105] “Nitro”, https://www.gonitro.com/free-pdf-software

[106] “Scientific and Technical Documentation Utility”, http://www.stdutility.com/stduviewer.html

[107] “New Features Summary, Adobe 2017”, https://helpx.adobe.com/acrobat/using/whats-new-acrobat-2017.html

[108] “CVE Details” https://www.cvedetails.com/index.php